VANESSA TEREZINHA ALES

O ALGORITMO *SEQUENTIAL MINIMAL OPTIMISATION* PARA RESOLUÇÃO DO PROBLEMA DE *SUPPORT VECTOR MACHINE*: UMA TÉCNICA PARA RECONHECIMENTO DE PADRÕES

Dissertação apresentada ao Curso de Pós-Graduação em Métodos Numéricos em Engenharia – Programação Matemática, Setores de Tecnologia e Ciências Exatas, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Mestre em Ciências.

Orientador: Prof. Dr. Celso Carnieri

CURITIBA
2008

TERMO DE APROVAÇÃO

VANESSA TEREZINHA ALES

O ALGORITMO *SEQUENTIAL MINIMAL OPTIMIZATION*
PARA RESOLUÇÃO DO PROBLEMA DE *SUPPORT VECTOR MACHINE*:
UMA TÉCNICA DE RECONHECIMENTO DE PADRÕES

Dissertação aprovada como requisito parcial para obtenção do grau de Mestre no Curso de Pós-Graduação em Métodos Numéricos em Engenharia, setores de Tecnologia e Ciências Exatas da Universidade Federal do Paraná, pela seguinte banca examinadora:

Orientador: _____
Prof. Dr. Celso Carnieri
Programa de Pós-Graduação em Métodos Numéricos em Engenharia - UFPR

_____
Prof. Dr. Arinei Carlos Lindbeck da Silva
Programa de Pós-Graduação em Métodos Numéricos em Engenharia - UFPR

_____
Prof. Dr. Alexandre Rasi Aoki
Programa de Pós Graduação em Engenharia Elétrica – UFPR e LACTEC

_____
Prof. Dr. Alessandro Lameiras Koerich
Programa de Pós Graduação em Informática Aplicada – PUCPR

Curitiba, 24 de Setembro de 2008

A Deus.

A meus pais Moises e Terezinha, por toda a dedicação, amor, carinho e incentivo, em todos os momentos.

Aos meus irmãos Wellington e William.

E a todos, que de muitas formas me incentivaram e ajudaram para que fosse possível a concretização deste trabalho.

Justamente quando você conseguir encontrar todas as respostas,
mudarão todas as perguntas.

Paulo Coelho

A vitória mais bela que se pode alcançar é vencer a si mesmo.

Santo Inácio de Loyola

AGRADECIMENTOS

A Deus, pela vida, benção, graças e proteção.

Aos meus pais, Moises e Terezinha, pelo constante apoio, incentivo, carinho e compreensão. Aos meus irmãos pela amizade e apoio.

Ao professor Celso Carnieri pela oportunidade, orientação, apoio, incentivo e paciência.

Ao professor Arinei Carlos Lindbeck da Silva pela oportunidade, apoio, incentivo, ensinamentos da linguagem *Visual Basic*.

Aos professores Sergio Scheer, Maria Teresinha Arns Steiner, Volmir Wilhelm, Luis Antonio Resende Santana, Neida Volpi, Liliana Cumin e Anselmo Chaves Neto, por contribuírem com a minha formação.

A secretária Maristela Brandil pelo apoio, carinho e amizade.

Ao amigo Vinícius, pelo incentivo e apoio, cuja amizade e companheirismo foram essenciais.

Aos colegas que se tornaram amigos: Ana Beatriz, Ricardo, Wyrllen, Alessandra, Francisco, Marcos Aurélio, Bernadete, Maiko, Roberta, Vanderlei e Crisiane cuja amizade e colaboração foram fundamentais.

Aos demais colegas que colaboraram no desenvolvimento da dissertação.

À Pró-Reitoria, pelo apoio financeiro com a bolsa de estudos.

A todos que me proporcionaram oportunidades para aquisição de conhecimentos e apoiaram meus estudos.

Vanessa Terezinha Ales

RESUMO

O presente trabalho tem por finalidade estudar o algoritmo *Sequential Minimal Optimization* (SMO) para resolução do problema quadrático do *Support Vector Machine* (SVM), uma técnica de reconhecimento de padrões. Diagnosticar doenças, automatizar processos de leitura de documentos, identificar pessoas, encontrar substâncias adulterantes em produtos são algumas aplicações de reconhecimento de padrões. Existem muitas técnicas de reconhecimento de padrões, porém, são limitadas a um conjunto de problemas. O SVM é uma técnica que utiliza a metodologia de aprendizagem supervisionada para o treinamento do processo de reconhecimento de padrões. A modelagem do problema do SVM envolve um problema quadrático convexo e, portanto, existe uma única solução, porém a busca dessa solução requer ferramentas complexas. A técnica de reconhecimento SVM tem como objetivo separar os padrões em classes distintas através de uma superfície separadora, esta com a maior distância possível das margens, que definem uma segurança na classificação de novos dados. As margens são determinadas pelos vetores suportes, definidos pelo valor de seu respectivo Multiplicador de Lagrange, proveniente da modelagem dual. O produto interno que aparece no problema dual pode ser substituído por uma função *kernel* que possibilita encontrar uma superfície separadora não linear, abrangendo mais aplicações. A utilização desta requer a escolha de uma função *kernel* adequada e a calibração dos parâmetros para que o processo apresente menos erros e generalize a situação. Para obtenção da solução do SVM utilizou-se o algoritmo SMO, desenvolvido por Platt, que escolhe dois pontos por iteração para fazer a otimização. Foram realizados treinamentos e testes em bancos de dados prontos resultando em processos de classificação muito bons se comparados com outras técnicas de reconhecimento de padrões.

Palavras chaves: *Support Vector Machine. Kernel.* Dualidade. Reconhecimento de padrões. Programação quadrática.

ABSTRACT

The purpose of this work is to study the Sequential Minimal Optimization (SMO) algorithm for solving the quadratic problem of Support Vector Machine (SVM), a technique for pattern recognition. Some applications of recognition are diagnosing diseases, automating the process of reading documents, identifying persons, finding adulterating substances in products. There are many techniques for achieving pattern recognition, each one limited to a set of problems. The SVM is a technique that uses the supervised learning methodology for training the process of pattern recognition. The model of SVM problem involves a convex quadratic problem and therefore, there exists a unique solution. However, for finding the solution it's necessary using complex tools. The technique of SVM has the objective of separating the patterns in distinct classes through a separating surface with the maximum distance from margin in order to obtain better results when classifying new data. The margins are determined by the support vectors, which are obtained calculating the Lagrange Multipliers in the dual model. The inner product that appears in the dual problem may be substituted by kernel function to find a non linear surface, boosting the possible applications. To obtain a good generalization and make fewer errors it is necessary to use an adequate kernel function and calibration of the parameters. To obtain the solution of the quadratic problem of SVM it was used the SMO algorithm developed by Platt, which chooses two points in each iteration for optimization. Many training experiments were made in data base resulting in good classification performance when compared with other pattern recognition methods.

Key works: Support Vector Machine. Kernel. Duality. Pattern recognition. Quadratic programming.

LISTA DE SIGLAS

AG – Algoritmos Genéticos
ESVM – *Evolutionary Support Vector Machine*
GSVM – *Generalized Support Vector Machine*
KDD – *Knowledge Discovery in Databases*
KKT – *Karush-Kuhn-Tucker*
LSVM – *Lagrangian Support Vector Machine*
LS-SVM – *Least Squares Support Vector Machine*
PSVM – *Proximal Support Vector Machine*
RNA – Redes Neurais Artificiais
RSVM – *Reduced Support Vector Machine*
SVM – *Support Vector Machine*
SVR – *Support Vector Regression*
SMO – *Sequential Minimal Optimization*
SSVM – *Smooth Support Vector Machine*
S³VM – *Semi-Supervised Support Vector Machine*
VC – Validação Cruzada
VS – Vetores suportes
VS-*bound* – Vetores suportes com valor máximo
ν-SVM – ν - *Support Vector Machine*

LISTA DE SÍMBOLOS

λ_i – Autovalor

b – *Bias*

y_i – Entrada

S – Conjunto de dados de treinamento

$\frac{\partial f}{\partial w}$ – Derivada parcial da função f em relação a variável w

n – Dimensão do espaço de entrada

N – Dimensão do espaço *feature*

E_i – Erro do ponto x^i no espaço *feature*

$\mathbb{R}^n$ – Espaço de Entrada

F – Espaço *feature*

$K(x, z)$ – Função *kernel* entre os pontos x e z

β_i – Índices na função da análise de discriminante

L – Lagrangeano Dual

W – Lagrangeano Primal

$\varphi : X \to F$ – Mapeamento para o espaço *feature*

γ – Margem funcional

α_i – Multiplicador de Lagrange do ponto x^i ou variáveis duais

$\|\cdot\|_p$ – Norma p

l – Número de pontos de treinamento

C – Parâmetro de ponderação dos erros

σ – Parâmetro do *kernel* gaussiano

p – Parâmetro do *kernel* polinomial

κ – Parâmetro do *kernel* sigmoidal

k – Parâmetro *kernel* polinomial não homogêneo e sigmoidal

β – Pesos da função discriminante

x^i – Ponto i do espaço de entrada

$\langle x \cdot z \rangle$ – Produto interno

ξ_i – Variável de folga do ponto x^i

w – Vetor de pesos

π_1, π_2 – Classes distintas

Σ_1, Σ_2 – Matrizes de covariância das classes π_1 e π_2

$\mu_1,\ \mu_2$ – Vetores de média das classes π_1 e π_2

$S_1,\ S_2$ – Matrizes de covariâncias amostrais das classes π_1 e π_2

$\overline{\mu_1}, \overline{\mu_2}$ – Vetores de média amostral das classes π_1 e π_2

$\text{sgn}\left(f\left(x\right)\right)$ – Sinal da função $f\left(x\right)$

$\left(\lambda_i, e_i\right)$ – Autovalor e autovetor

LISTA DE TABELAS

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

O reconhecimento de padrões é um processo de classificação de dados, informações, imagens e objetos. O cérebro humano realiza esse processo instantaneamente, mas computacionalmente, esse processo não é tão rápido, nem tão preciso. Em virtude disso, muitas pesquisas foram e continuam sendo realizadas nessa área. O primeiro algoritmo de reconhecimento foi elaborado por Fisher, em 1936, denominado de Discriminante de Fisher, o qual utiliza conceitos estatísticos.

Mais tarde, Rosemblat criou um algoritmo utilizando o conceito de aprendizagem, denominado de Perceptron de Rosemblat, dando início ao conceito de Redes Neurais Artificiais, que se baseiam no funcionamento do cérebro humano.

Outras técnicas e algoritmos foram sendo criados, para obtenção de resultados melhores e mais precisos. Cada técnica limitava as condições para geração do algoritmo e isso gerava resoluções boas apenas àqueles conjuntos que continham as mesmas limitações.

O *Support Vector Machine* (SVM) é uma técnica de reconhecimento de padrões baseada na metodologia de aprendizagem, gerando resultados robustos.

Desenvolvido durante três décadas, passou por melhoramentos importantes. Inicialmente a técnica classificava apenas pontos linearmente separáveis, separando as classes com a máxima distância entre eles. Em virtude disso, a probabilidade de classificar um ponto incorretamente tornava-se mínima.

Posteriormente, introduziu-se a função *kernel*, uma forma de mapear os pontos em uma dimensão mais elevada, onde os pontos seriam linearmente separáveis. Outra modificação importante foi a introdução de variáveis de folga, que permitiam classificar pontos como errados no treinamento, maximizando a distância e percebendo dados incorretos.

A junção de variáveis de folga com a função *kernel* possibilitou a classificação de dados não separáveis linearmente, construindo superfícies de separação.

Os resultados do SVM, como técnica de reconhecimento de padrões, mostraram-se satisfatórios, obtidos nas mais variadas áreas, porém, sua resolução é complexa, tendo permitido o surgimento de heurísticas e variações da técnica. A complexidade do algoritmo de aprendizagem do SVM está relacionada ao problema quadrático envolvido na sua formulação, cuja resolução exige o uso de ferramentas complexas.

A principal característica do SVM é a detecção automática de vetores suportes, que correspondem a uma fração do conjunto de treinamento. Os vetores suportes são os únicos vetores (dados) relevantes para a construção da superfície de separação, que pode classificar dados novos.

Nem sempre os vetores suportes escolhidos construirão a melhor superfície de separação, dependendo da importância que os dados de treinamento possuem em relação aos demais pontos.

1.1 OBJETIVOS GERAIS

O objetivo principal deste trabalho é estudar as técnicas de reconhecimento de padrões, principalmente o método de *Support Vector Machine* (SVM) e suas variações. Mais especificamente, os objetivos são estudar e implementar computacionalmente o SVM, utilizando a ferramenta computacional *Lingo*, e o *Sequential Minimal Optimization* (SMO), um algoritmo do SVM, e analisar os resultados.

Uma aplicação, em especial, será realizada utilizando o banco de dados da Professora Maria Teresinha Steiner Arns, dados de pacientes com icterícia. Essa aplicação já foi estudada, possui resultados utilizando outras técnicas que podem ser comparados com o método do SVM.

Em virtude disso, possibilitar o estudo de soluções para a técnica do SVM, a qual pode ser base para outras pesquisas.

1.2 IMPORTÂNCIA DO TRABALHO

O reconhecimento de padrões torna-se cada vez mais importante em diversas áreas, principalmente na medicina e na biologia, em virtude de haver uma grande quantidade de problemas reais, para os quais não existem soluções exatas e as técnicas de reconhecimento de padrões mais conhecidas serem limitadas para a resolução de um subconjunto de problemas. Buscam-se então técnicas com resultados melhores e que consigam resolver um maior número de problemas.

O método do SVM tem tido destaque na literatura, pelo sucesso dos resultados e as mais diversas aplicações, tornando-se atrativo para obtenção de melhores resultados.

1.3 LIMITAÇÕES DO TRABALHO

Neste trabalho, contempla-se apenas a classificação de dados com saída binária, somente referenciando a classificação de saídas múltiplas.

Estudo limitado sobre o Teorema de *Mercer* e as condições de Karush-Kuhn-Tucker, pois envolvem assuntos relacionados à Análise Funcional e Otimização.

1.4 JUSTIFICATIVA

O modelo de otimização gerado pelo SVM envolve em sua formulação um problema quadrático. Este garante a existência de uma solução ótima. A desvantagem do problema quadrático é necessitar de ferramentas complexas para resolução, como por exemplo, a ferramenta computacional Lingo, que está limitada a determinado número de dados.

Em virtude dessa dificuldade, optou-se em utilizar o algoritmo SMO, que possibilita encontrar a solução com mais agilidade.

1.5 ESTRUTURA DA DISSERTAÇÃO

A dissertação está organizada em sete capítulos, incluindo este. Neste primeiro capítulo encontra-se esta introdução com tema, objetivos e justificativa do trabalho. No segundo capítulo, a revisão de literatura envolvendo reconhecimento de padrões e técnicas utilizadas para classificação de dados; conceitos e teoremas fundamentais no desenvolvimento da técnica do SVM.

O conceito do SVM está descrito em detalhes no terceiro capítulo. No capítulo quatro encontra-se o algoritmo para resolução do mesmo utilizando uma heurística, bem como encontram-se a metodologia utilizada, os dados e o *software* desenvolvido. O quinto capítulo contempla aplicações utilizando o SVM e suas variações. No capítulo seis está a análise dos resultados.

Finalmente no capítulo sete, as conclusões e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

O objetivo deste capítulo é compreender a importância do processo de reconhecimento de padrões e suas aplicações em problemas reais. O processo KDD e as técnicas utilizadas em separar dados em dois grupos tais como, análise de discriminante, regressão logística, redes neurais, algoritmos genéticos e *Support Vector Machine*.

Este capítulo também contempla definições e teoremas necessários para desenvolvimento do *Support Vector Machine*. Esse desenvolvimento contempla pesquisa operacional, programação não linear, principalmente programação quadrática, dualidade e o lagrangeano.

2.1 RECONHECIMENTO DE PADRÕES

Padrões são entidades, às quais podem ser atribuídas características (Tan, Steinbach, Kumar, 2005).

Um padrão pode ser um lápis, por exemplo, algumas de suas características são: um grafite ou giz colorido em formato de prima ou cilindro revestido por um invólucro de madeira, uma extremidade apontada utilizada para escrever ou desenhar, comprimento de aproximadamente quinze centímetros, cores diversificadas e inúmeras outras características. Conforme pode ser analisado na figura 1.

FIGURA 1 – MODELO DE LÁPIS
FONTE: Internet[1] (2008)

A habilidade do ser humano em reconhecer e classificar objetos sempre impressionou e continua a impressionar os cientistas. Se juntarmos a capacidade do ser

[1] Figura retirada do pagina http://www.olhao.com.br/educacao 12112007191825.shtml em 03/03/2008.

humano com a velocidade do computador pode ser agilizado o processo de reconhecimento.

Desde os primórdios da computação, a tarefa de programar algoritmos imitando a capacidade humana, tem-se apresentado como a tarefa mais intrigante e desafiadora. Ainda não foi criado *software* tão eficiente em reconhecimento como o cérebro humano para tarefas como comparação visual.

O reconhecimento de padrões é uma ciência que trata das técnicas de classificação de padrões através de suas características ou propriedades, conforme ilustra a figura 2:

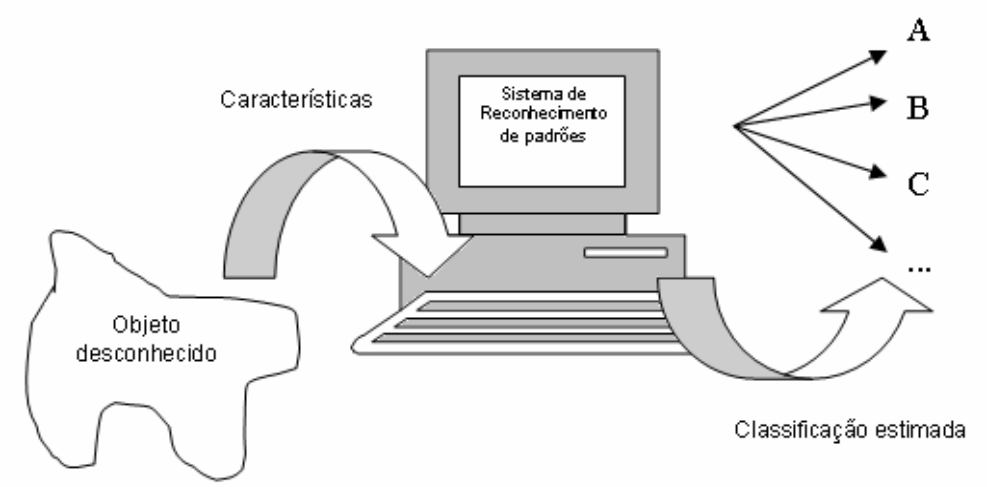

FIGURA 2 – PROCESSO DE RECONHECIMENTO DE PADRÕES
FONTE: A autora (2008)

Um método de reconhecimento de padrões deve basear-se no conhecimento extraído dos exemplos de uma base de dados e na classificação de um exemplo novo, desconhecido até então, ao padrão que mais reflete as suas características (CARVALHO e BRAGA, 2004).

Desenvolvem-se cada vez mais poderosos métodos de reconhecimento de padrões, principalmente para tarefas como o reconhecimento de dígitos, reconhecimento de faces, falhas em equipamentos, tendências financeiras e muitos outros.

Para realizar o reconhecimento de padrões é necessário realizar o processo de selecionar os dados de entrada, extrair as características e classificar.

A filtragem da entrada de dados tem o objetivo de eliminar dados desnecessários ou distorcidos fazendo com que a entrada apresente apenas dados relevantes para o

reconhecimento de padrões. A extração de características consiste da análise dos dados de entrada com o objetivo de extrair características úteis para o processo de reconhecimento. O estágio final é a classificação, onde através da análise das características da entrada de dados o padrão analisado é declarado como pertencente a um determinado conjunto ou não.

Para realizar a classificação de padrões existem alguns desafios propostos ao algoritmo, podendo-se citar a robustez e a estabilidade estatística como os mais importantes (SHAWE-TAYLOR e CRISTIANINI, 2004).

A ferramenta deve ser robusta, não sendo sensível a dados que apresentem ruídos, ou seja, valores afetados por leituras de medidas incorretas. Outra característica, talvez a mais importante, é que os padrões devem ser reais, e não uma relação acidental a um conjunto limitado de dados, para obter qualidade na generalização do problema.

Muitas vezes na aprendizagem, o problema não está generalizado, ou seja, é um caso específico a poucos dados, pode ocorrer *underfitting* ou *overfitting*, isto é, não houve ajuste suficiente ou exagerou-se no ajuste do modelo, respectivamente (SHAWE-TAYLOR e CRISTIANINI, 2004).

Para classificar os padrões existem vários métodos: Redes Neurais Artificiais, Análise Multivariada, Algoritmos Genéticos, *Support Vector Machine* (SVM) entre outros.

2.1.1 Exemplos de reconhecimento de padrões

As ferramentas de reconhecimento de padrões possuem as mais variadas aplicações.

O monitoramento de imagens obtidas através de satélites e de aviões possibilita analisar o uso das terras (BITTENOURT, 2001) e fazer a mosaicagem[2] (SILVA e CANDEIAS, 2003).

Na medicina, o processamento de imagens e sinais possibilita o diagnóstico de câncer de mama (MANGASARIAN, SETIONO E WOLBERG, 1990), fibrilação atrial[3]

[2] Mosaicagem é um termo utilizado para definir a junção de mapas, ou seja, mapas separados, porém adjacentes, são agrupados em um único mapa.
[3] Fibrilação atrial é catacterizada pela propagação de ondas em diferentes direções, causando atividade atrial desorganizada.

(PILLA e LOPES, 1999), diabetes mellitus (STOEAN, 2005), síndrome genética (DAVID e LERNER, 2005) e detectar os limites de um melanoma[4] com precisão.

Na biologia, as técnicas de reconhecimento de padrões podem ser utilizadas para analisar a qualidade ambiental de riachos (FERNANDES, 2006), quantificar os adulterantes do leite (FERRÃO et al, 2007) entre outras.

A biometria é o estudo estatístico das características físicas e comportamentais das pessoas como forma de identificá-las unicamente. Atualmente, a biometria é usada na identificação criminal, controle de ponto e chave de acesso em instalações de segurança e a contas bancárias. Algumas das técnicas de reconhecimento de características físicas utilizadas são: a voz, a íris, a retina, a face (LI, GONG e LIDDELL, 2000), expressões faciais (FAN et al, 2005), as impressões digitais, a assinatura (ÖZGÜNDÜZ, SENTÜRK e KARSLIGIL, 2005), veias e geometria das mãos.

Na indústria, o reconhecimento pode ser utilizado para analisar os transformadores de potência (PAIXÃO, 2006), a qualidade do óleo isolante dos mesmos (SOUZA, 2008) e outros.

O reconhecimento de padrões é utilizado no processamento de documentos como leitura de crachás com código de barras e reconhecimento dos caracteres e algarismos em placas de veículos.

2.2 KDD

O termo KDD do inglês “Knowledge Discovery in Databases” que significa “Descoberta de Conhecimento em Bases de Dados”, é um processo não trivial de descoberta de padrões válidos, novos, úteis e acessíveis (FAYYAD et al.,1996). De acordo com os autores, o processo KDD é composto por cinco etapas: Seleção, Pré-Processamento, Transformação, Mineração e Interpretação dos Dados, como ilustrado na figura 3.

[4] Melanoma é o câncer de pele.

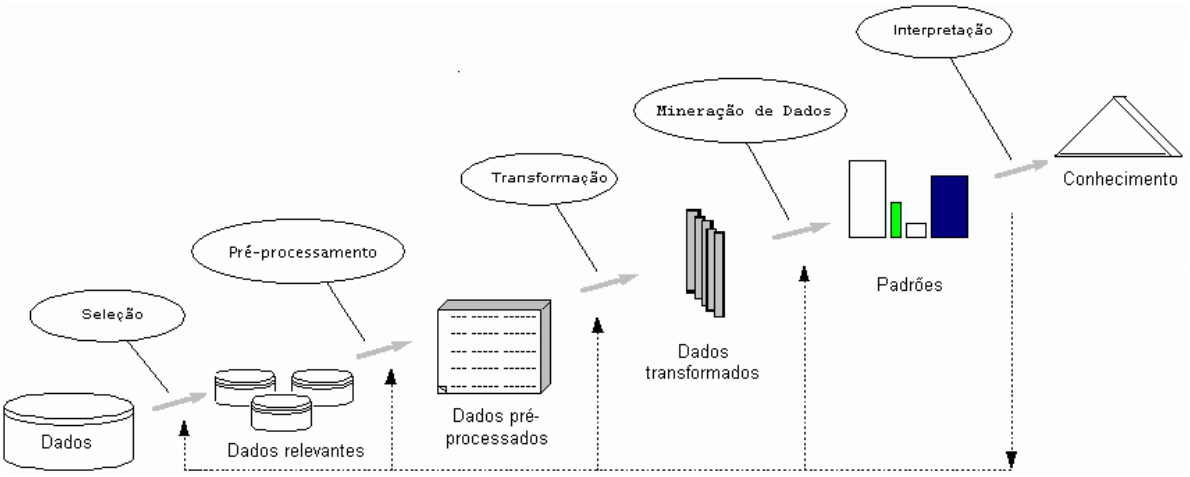

FIGURA 3 – PROCESSO DE KDD
FONTE: Internet[5] (2008)

As fases do KDD, ilustradas na figura 3, são descritas abaixo:

1. Seleção: é a etapa que consiste na análise dos dados existentes e na seleção daqueles a serem utilizados na busca por padrões e na geração de conhecimento novo.

2. Pré-processamento: consiste no tratamento e na preparação dos dados para uso dos algoritmos, como identificar e retirar valores inválidos, inconsistentes ou redundantes.

3. Transformação: consiste em aplicar, quando necessário, alguma transformação linear ou não linear nos dados, de forma a encontrar aqueles mais relevantes para o problema em estudo. Geralmente, são aplicadas técnicas de redução de dimensionalidade e de projeção dos dados.

4. *Data Mining*: consiste na busca por padrões através da aplicação de algoritmos e técnicas computacionais específicas.

5. Interpretação: consiste na análise dos resultados e na geração de conhecimento pela interpretação e utilização dos resultados em benefício do reconhecimento de padrões.

O processo de KDD é interativo e, geralmente, envolve diversos laços de repetição dentro de uma mesma etapa e também entre fases, até que um resultado útil seja alcançado.

[5] Figura retirada do site: http://www.javaportal.it/portal em 21/07/2008.

2.3 *DATA MINING*

A etapa *Data Mining* (Mineração de Dados) é o núcleo do processo, onde são aplicados os algoritmos para extrair padrões dos dados. (FAYYAD et al., 1996).

Mineração de Dados é a extração de informação ou conhecimento útil nos dados.

ZAKI et al. (2007) definem Mineração de Dados como um processo de descoberta automática de novos e compreensíveis modelos e padrões em grandes quantidades de dados.

A Mineração de Dados tem como base a Estatística, a Inteligência Artificial, a Aprendizagem de Máquinas, o Reconhecimento de Padrões e os Sistemas de Bases de Dados. É adequada à análise de grandes quantidades de dados, dados com alta dimensionalidade e com natureza distribuída e heterogênea (TAN et al., 2005).

As tarefas de Mineração de Dados são preditivas ou descritivas. As preditivas usam algumas variáveis para prever valores desconhecidos ou futuros de outras variáveis, enquanto que as descritivas encontram padrões para descrever os dados (FAYYAD et al., 1996). As principais tarefas de Mineração de Dados estão relacionadas à Classificação, Associação e Agrupamento de padrões.

Na Classificação cada padrão contém um conjunto de atributos e um dos atributos é denominado classe. O objetivo da classificação é encontrar um modelo para predição da classe como função dos outros atributos (TAN et al., 2005). A regressão é um caso particular da classificação, já que seu objetivo é encontrar um modelo para predição de um atributo contínuo como função dos outros atributos.

Já na Associação, o objetivo é produzir regras de dependência que irão predizer a ocorrência de um atributo baseado na ocorrência de outros atributos (TAN et al., 2005).

O Agrupamento ou Segmentação (também conhecido pelo termo em inglês “Clustering”) procura grupos de padrões tal que padrões em um grupo são mais similares uns aos outros e dissimilares a padrões em outros grupos (TAN et al., 2005).

O objetivo deste trabalho é reconhecer padrões através da classificação, e para isto, estudam-se alguns métodos tais como Análise Multivariada, Redes Neurais Artificiais, Algoritmos Genéticos, *Support Vector Machine* e outros.

2.3.1 Análise Multivariada

A Análise Multivariada é a área estatística que trabalha com várias variáveis simultaneamente, proporcionando a classificação numa situação real, que, normalmente, envolve várias características.

Dentre as diversas técnicas estatísticas multivariadas, destacam-se as utilizadas para reconhecimento de padrões, como a Análise de Discriminante, que utiliza a Função Discriminante Linear de Fischer e a Regressão Logística.

2.3.1.1 Análise de Discriminante de Fischer

A Análise de Discriminante é uma técnica estatística multivariada usada na resolução de problemas que envolvem a separação de conjuntos distintos de observações e a alocação de novas observações em um dos conjuntos (GUIMARÃES e CHAVES NETO, 2006).

O objetivo da Análise de Discriminante é obter uma regra de classificação que minimize a probabilidade de classificação errônea, utilizando variáveis que permitam salientar as diferenças entre os grupos, através de observações em uma ou mais classes pré-determinadas (CUNICO, 2005).

Para usar esta técnica, utiliza-se a Função Discriminante de Fischer.

Para construir a função discriminante para duas classes π_1 e π_2, considera-se o vetor $X = \left(x_1, x_2, ..., x_p\right)$, representando as p características dessas classes e Y uma variável dependente, a resposta. Uma função discriminante linear tem a forma:

$$Y = \beta_0 + \sum_{i=1}^{p-1} \beta_i X_i \qquad (2.1)$$

Em problemas envolvendo duas classes, Y é uma variável dicotômica, que representa a qual classe a amostra pertence. Geralmente, são atribuídos à resposta os valores 1 e 2. Os valores β_i, com $i = 0,1,...,p$ são os coeficientes da função nas variáveis X_i, $i = 0,1,...,p$, com $X_0 = 1$. O valor de Y funciona como um escore de

classificação, pois a função retorna um valor para cada novo padrão m_i, que é um vetor com as medidas de uma nova observação.

A função discriminante linear transforma a observação multivariada X, em uma observação univariada Y (escore), tal que os escores obtidos separem ao máximo as classes π_1 e π_2.

A função que expressa a regra de classificação é dada pela equação:

$$y = (\mu_1 - \mu_2)' \Sigma^{-1} X \tag{2.2}$$

onde:

μ_1, μ_2 são os vetores das médias das classes π_1 e π_2

Σ é a matriz de covariâncias comum a ambas as classes,

ou seja, $\Sigma_1 = \Sigma_2$

Caso $\Sigma_1 \neq \Sigma_2$, a regra de classificação torna-se mais complicada, o que pode ser visto com detalhes em Johnson e Wichern (1998).

A amostra X_j é alocada na população π_1 se $y_j - m \geq 0$, e X_j é alocada na classe π_2, caso $y_j - m < 0$, onde m é o valor de corte.

Na realidade, os parâmetros μ_1, μ_2 e Σ não são conhecidos. Portanto, trabalha-se com estimadores $\overline{\mu_1}$, $\overline{\mu_2}$ e S_p, obtidos de amostras aleatórias das classes π_1 e π_2, com tamanhos n_1 e n_2, respectivamente. O estimador S_p tem a expressão:

$$S_p = \frac{1}{n_1 + n_2 - 2}\left[(n_1 - 1)S_1 + (n_2 - 1)S_2\right] \tag{2.3}$$

onde:

S_1 e S_2 são as matrizes de covariâncias amostrais.

Assim, obtem-se a Função Discriminante Linear de Fischer Amostral cuja expressão é:

$$\hat{y} = \left(\overline{X_1} - \overline{X_2}\right)' S_p^{-1} X \tag{2.4}$$

O valor de corte m é estimado por: $\widehat{m} = \frac{1}{2}\left(\overline{y_1} + \overline{y_2}\right)$, onde $\overline{y_1}$ e $\overline{y_2}$ são as médias dos escores para π_1 e π_2.

Portanto, a regra de classificação é dada por: alocar X_j na classe π_1 se $y_j = \left(\overline{X_1} - \overline{X_2}\right)' S_p^{-1} X_j \geq \widehat{m}$ e alocar X_j na classe π_2 se $y_j = \left(\overline{X_1} - \overline{X_2}\right)' S_p^{-1} X_j < \widehat{m}$.

2.3.1.2 Regressão Logística

A técnica de Regressão Logística, que trata do ajuste do Modelo Logístico, geralmente, é utilizada para tratar de problemas relacionados a dados dicotômicos em várias áreas de conhecimento, sendo interessante conhecer a probabilidade de um indivíduo pertencer a um determinado grupo (CUNICO, 2005).

Existem algumas razões que tornam a Regressão Logística mais atraente do que Discriminante de Fisher, entre elas o fato da Regressão Logística ser mais robusta quando os pressupostos não são conhecidos. Essa técnica estima a probabilidade de um evento ocorrer ou não, podendo ser classificado em dois grupos (HAIR et al, 1998).

O modelo de regressão logística é baseado na função sigmóide, dada por:

$$f(y) = \frac{e^y}{1 + e^y} \tag{2.5}$$

onde:

$$y = \beta_0 + \sum_{i=1}^{p-1} \beta_i X_i$$

Onde se considera o vetor $X = \left(x_1, x_2, ..., x_p\right)$, representando as p características dessas classes e y representando um índice que combina a contribuição dos fatores de risco e $f(y)$ representa a probabilidade de que o evento ocorra.

Portanto, observa-se que a função logística varia entre 0 e 1, $0 \leq f(y) \leq 1$ para $y \in (-\infty, +\infty)$, razão pela qual o modelo descreve uma variável dicotômica.

A estimação dos coeficientes $\beta_0, \beta_1, ..., \beta_{p-1}$ é feita pelo método da Máxima Verossimilhança ou dos Mínimos Quadrados, e o cálculo segue o Método de Newton (CUNICO, 2005).

O método da Máxima Verossimilhança é sempre mais indicado por possuir propriedades ótimas. Desta maneira, têm-se as estimativas dos parâmetros dadas por $\hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_{p-1}$, que por sua vez alimentam o modelo estimado, dado por:

$$\hat{f}(y) = \frac{e^{\hat{y}}}{1 + e^{\hat{y}}} \tag{2.6}$$

onde:

$$y = \hat{\beta}_0 + \sum_{i=1}^{p-1} \hat{\beta}_i X_i$$

Para classificar uma amostra basta substituí-la no modelo e encontrar o valor estimado da função $\hat{f}(y)$, que logicamente será um valor entre 0 e 1, de acordo com esse valor classificar a amostra em sua respectiva categoria dicotômica.

Uma característica, que torna esse modelo muito utilizado, é o fato de possuir a curva logística em forma de “S”, indicando que o efeito de y na função $f(y)$ é o menor possível até um nível, depois aumenta rapidamente até algum outro nível, no qual a função volta a crescer lentamente.

2.3.2 Redes Neurais Artificiais

As redes neurais são inspiradas no sistema nervoso biológico[6]. O sistema nervoso humano é constituído de aproximadamente duzentos bilhões de células e possui algumas características principais: é uma rede altamente conectada, muitos neurônios operam ao mesmo tempo, sendo a informação distribuída, admite falha, entre outras características (STEINER, 1995).

As células, chamadas neurônios, conforme ilustra a figura 4, são capazes de se interligar a fim de realizar tarefas de reconhecimento de imagens, padrões e outros. Tem

[6] O neurônio, célula nervosa, foi identificado anatomicamente e descrito com detalhes pelo neurologista espanhol Ramòn y Cajal, em 1894.

a capacidade de aprender através de estímulos do meio ambiente, adaptando-se às mudanças do mesmo.

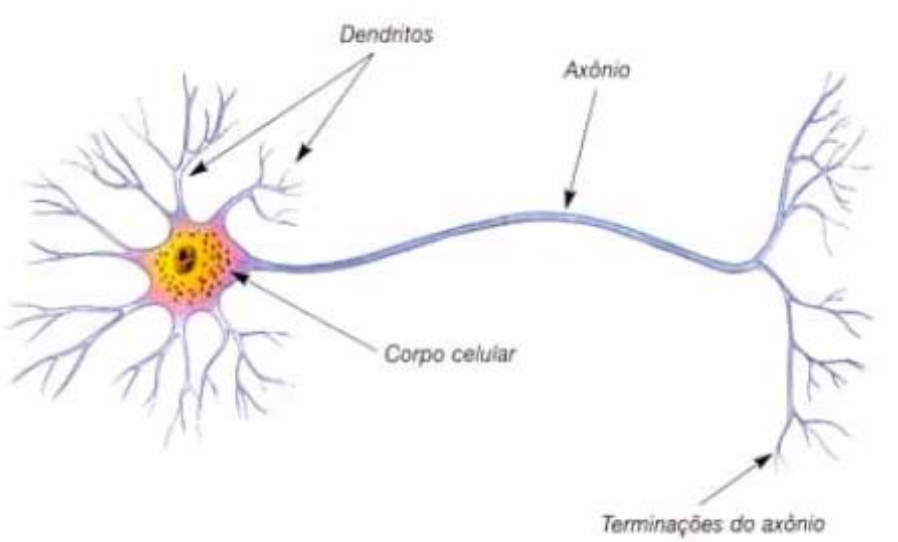

FIGURA 4 – NEURÔNIO BIOLÓGICO[7]
FONTE: Internet[8](2008)

Portanto, tornou-se interessante compreender a funcionamento do cérebro humano e reproduzir algumas de suas características e posteriormente utilizá-las no reconhecimento de padrões.

Ao tentar imitar a capacidade de aprendizagem dos neurônios, em 1943, McCulloch e Pitts propuseram um modelo para a célula nervosa e criaram um neurônio artificial, mostrando que uma coleção de neurônios era capaz de calcular certas funções lógicas. Em 1949, Hebb apontou o significado das conexões entre os neurônios, conhecidas por sinapses, para o processo de aprendizagem e desenvolveu uma regra básica de aprendizagem. Propôs que a força das sinapses é proporcional às ativações dos neurônios (STEINER, 1995).

Em 1959, Rosenblatt descreveu o primeiro modelo de rede neural, denominado Perceptron. Esse modelo podia aprender funções lógicas, arranjando os neurônios em uma rede e modificando as conexões entre as sinapses. Em 1962, Widrow desenvolveu um outro processador para redes neurais, o Adaline, o qual possui uma estratégia para

[7] O neurônio é delimitado por uma fina membrana celular, que além da função biológica normal, possui outras propriedades que são fundamentais para o funcionamento elétrico da célula. O corpo do neurônio é responsável por coletar e combinar informações vindas de outros neurônios. Os dendritos recebem os estímulos transmitidos pelos outros neurônios. O axônio é responsável em transmitir os estímulos para outras células e pode atingir até alguns metros. (Cunico, 2005)
[8] Figura retirada do site: http://dukkha.weblogger.terra.com.br em 13/03/2008

o processamento. Em 1974, Werbos lançou o algoritmo *Back–Propagation,* que permitiu a rede neural com múltiplas camadas com capacidade de aprendizado, obtendo um maior progresso em relação ao Perceptron de Rosemblatt (STEINER, 1995).

Um esquema do neurônio artificial é dado pela figura 5.

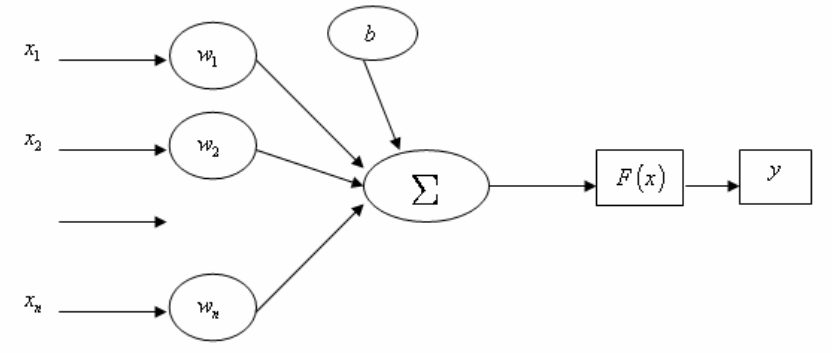

FIGURA 5 – NEURÔNIO ARTIFICIAL
FONTE: Steiner (1995)

O vetor x representa o vetor dos dados de entrada, sendo n o número de dados, e w é um vetor de pesos. É realizado o produto $p = xw$ e aplicado aos canais de entrada do neurônio.

A soma de todas as entradas ponderadas é processada por uma função de ativação, $F(x)$ para resultar o sinal da saída do neurônio, dado por:

$$y = \begin{cases} +1 \text{ ,se } F\left(\sum_{i=0}^{n-1} x_i w_i + b\right) > 0 \\ \\ -1 \text{ ,se } F\left(\sum_{i=0}^{n-1} x_i w_i + b\right) < 0 \end{cases} \tag{2.7}$$

O parâmetro b é denominado *bias* ou limiar, sua função é aumentar o número de graus de liberdade disponíveis no modelo, permitindo que a rede neural tenha maior capacidade de se ajustar ao mesmo.

Este modelo é baseado no neurônio biológico, porém o neurônio biológico tem saída binária, e o neurônio artificial pode ter saída binária ou contínua, em virtude de depender da função de ativação $F(x)$.

A função de ativação pode ser uma função matemática e geralmente são utilizadas funções lineares, função degrau, funções sigmóides ou logísticas ou ainda função tangente hiperbólica, pois podem ter saída no intervalo $[-1,+1]$.

Uma rede neural obedece a um processo de aprendizagem, sendo que o treinamento pode ser supervisionado e não supervisionado. O treinamento supervisionado ocorre quando a rede é treinada usando os dados de entrada e seus respectivos dados de saída. No treinamento não supervisionado somente os dados de entrada são utilizados para a aprendizagem. Um algoritmo foi proposto por Kohonen, em 1984, extraindo propriedades estatísticas do conjunto de treinamento.

Os principais modelos de Redes Neurais Artificiais (RNA) são o Perceptron, Redes Lineares e Redes de Múltiplas Camadas (BISHOP, 1995).

A Rede Perceptron possui uma única camada de i neurônios conectados a n entradas através do conjunto de pesos w. Os Perceptrons são treinados utilizando exemplos corretos, ou seja, um treinamento supervisionado. O erro, diferença entre o valor de saída e a saída desejada, é utilizado para atualizar os pesos w e o *bias*. Podem ocorrer respostas distintas, dependendo do valor inicial dos pesos w (STEINER, 1995).

Porém, existem limitações, das quais se destacam: depender de um vetor de pesos inicial; possuir saída binária; somente classificar conjuntos linearmente separáveis e o processo de convergência ser muito lento, dependendo do número de dados de cada conjunto a ser classificado.

As Redes Lineares permitem que as saídas sejam quaisquer valores entre 0 e 1. Utilizando a regra de aprendizagem de Widrow-Hoff, também conhecida como Mínimos Quadrados, a atualização dos valores dos pesos e do *bias* depende do valor da magnitude dos erros (STEINER, 1995).

As Redes de Múltiplas Camadas ou Redes *Feed-Forward* são uma generalização da regra de Widrow-Hoff para múltiplas camadas e funções de transferência não lineares. A dificuldade desta rede é encontrar o número adequado de neurônios na camada escondida. Para treinamento da rede de Múltiplas Camadas é utilizado o algoritmo *Back-Propagation* e a função de transferência Sigmoidal (STEINER, 1995).

2.3.3 Algoritmos Genéticos

Em 1859, Charles Darwin[9] expôs suas idéias a respeito da evolução dos seres vivos. Porém, não obteve muito sucesso, pois na época os cientistas acreditavam na Teoria do Criacionismo[10] ou na Teoria da Geração Espontânea[11].

A Teoria da Evolução afirma que todos os indivíduos dentro de um determinado sistema competem entre si pelos recursos limitados. Aqueles que têm menos êxito na busca desses recursos acabam sendo eliminados naturalmente, a essa eliminação espontânea é conhecida como seleção natural. Em virtude disso, os indivíduos mais adaptados ao ambiente sobrevivem, gerando populações mais adaptadas (HUBNER, 2008).

Apenas no século XX, com o redescobrimento dos trabalhos de Mendel[12] e com o aprofundamento do conceito de gene, é que se ampliam às idéias de Darwin sobre a evolução. A evolução estava relacionada com a genética.

Em 1975, John Holland[13], da Universidade de Michigan, desenvolveu um algoritmo de otimização inspirado na genética e nas Teorias de Evolução de Darwin.

Mais tarde, David Goldberg publicou um livro tutorial sobre algoritmos genéticos e realizaram-se Conferências Internacionais, que ampliaram a área dos mesmos.

Os Algoritmos Genéticos (AG) pertencem à classe dos algoritmos probabilísticos de busca de otimização, mas não são aleatórios, pois dirigem a busca para regiões onde seja provável que os pontos estejam ótimos.

Na metodologia dos Algoritmos Genéticos representa-se cada indivíduo como um cromossomo, com uma seqüência de genes. Cada gene representa uma característica do indivíduo, que é escrita como uma série de números, geralmente usando-se a representação binária.

Na figura 6, a representação de um cromossomo na biologia e no Algoritmo Genético:

---

[9] Charles Robert Darwin (1809-1882), naturalista inglês, em 1859 publicou o livro sobre a origem das espécies por meio de seleção natural, concluindo que as variações adaptativas ou favoráveis contribuem para a adaptação do indivíduo no meio em que vive, enquanto as variações desfavoráveis tendem a extinção da espécie.

[10] Teoria do Criacionismo: “Deus criou o universo da forma que ele é”.

[11] Teoria da Geração Espontânea: “A vida surge dos elementos presentes na natureza”.

[12] Gregor Johann Mendel (1822-1884), monge agostiniano, botânico e meteorologista austríaco, desenvolveu experimentos sobre a hereditariedade vegetal, com ervilhas-de-cheiro (*Pisum sativum*), que permitiram a descoberta das Leis de Mendel, que constituem o alicerce da genética, conhecidas também como Leis Básicas da Genética.

[13] John Holland publicou o livro *Adaptation in Natural and Artificial Systems*, mas este foi pouco divulgado, devido à notação pouco criteriosa e excessivamente complexa.

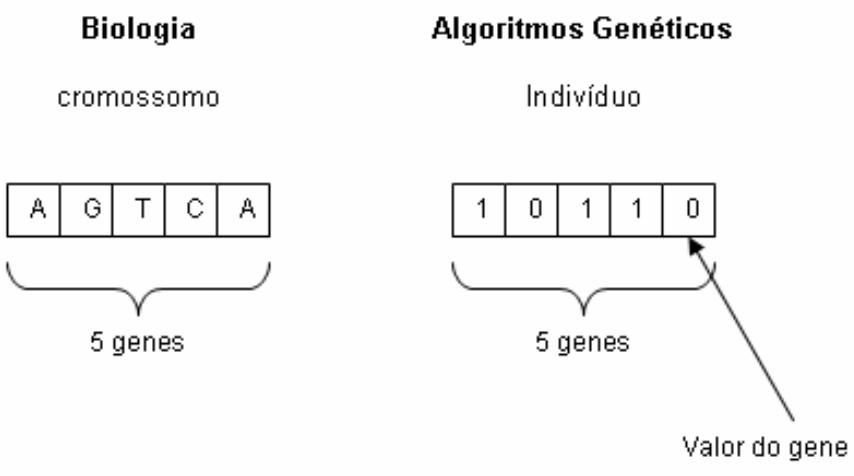

FIGURA 6 – REPRESENTAÇÃO DO CROMOSSOMO NA BIOLOGIA E NO AG
FONTE: Correa (2000)

Para iniciar o algoritmo, gera-se um conjunto de soluções-candidatas, uma população inicial aleatória, e através de certas iterações, conhecidas como gerações, toda a população será modificada para uma população ótima.

As iterações ocorrem com a avaliação da função *fitness* (adaptabilidade ou aptidão) dos cromossomos, através da probabilidade de seleção de um indivíduo, geralmente dada por:

$$P_{sel\_x_k} = \frac{f(x_k)}{\sum_{i=1}^{n} f(x_i)} \tag{2.8}$$

A equação 2.8 é dada para problemas de maximização, sendo n o número de indivíduos da população, $f(x)$ o valor da função *fitness* e x_k as soluções candidatas.

Assim, indivíduos com pouca probabilidade de adaptação tendem a ser extintos e os indivíduos mais adaptados possuem maiores chances de sobrevivência.

As operações que definem novas gerações são reproduções, cruzamentos e mutações. As reproduções consistem em copiar integralmente um indivíduo selecionado para a próxima geração (Figura 7).

FIGURA 7 – ESQUEMA DE REPRODUÇÃO NOS AGS
FONTE: Correa (2000)

Os cruzamentos imitam a reprodução sexuada, onde dois pais são selecionados e trocam entre si parte de seus cromossomos, gerando dois filhos (Figura 8).

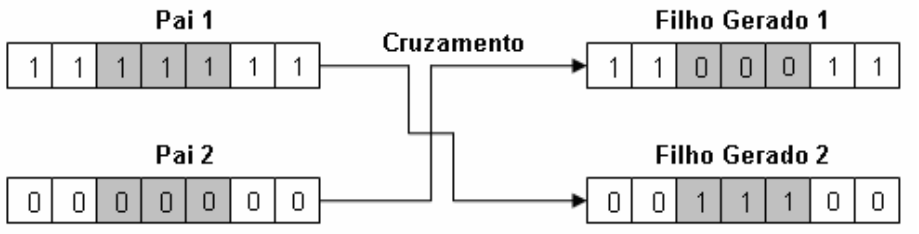

FIGURA 8 – ESQUEMA DE CRUZAMENTO NOS AGS
FONTE: Correa (2000)

Para diversificar as gerações são feitas mutações, que consiste em sortear um gene e alterá-lo (Figura 9).

FIGURA 9 – ESQUEMA DE MUTAÇÃO NOS AGS
FONTE: Correa (2000)

Normalmente, o critério de parada dos Algoritmos Genéticos é o número de gerações (CORREA, 2000).

Os Algoritmos Genéticos geralmente são aplicados a problemas com grande região de busca, sendo mais robusto do que outros métodos de direção de busca (NGUYEN et al, 2006).

A estratégia utilizada pelos Algoritmos Genéticos emprega mecanismos de busca que combinam escolhas aleatórias com o conhecimento obtido nos resultados anteriores. Evitando paradas prematuras em ótimos locais e proporcionando melhores soluções para o problema. Nem sempre os Algoritmos Genéticos garantem a solução ótima, mas, geralmente obtém soluções próximas ou aceitáveis.

Stoean (2005) propôs um modelo de reconhecimento de padrões utilizando a técnica dos Algoritmos Genéticos. O objetivo do reconhecimento era identificar a presença de *diabetes mellitus* ou não, nos pacientes. Os resultados foram ótimos, comparados com outros métodos.

Os Algoritmos Genéticos também são utilizados para balanceamento de pesos, como, por exemplo, no trabalho de Pilla e Lopes (1999), para encontrar o conjunto de pesos otimizados das redes *neurofuzzy*.

2.4 SUPPORT VECTOR MACHINE

O *Support Vector Machine* (SVM) é um sistema de aprendizagem muito utilizado tanto em aplicações de classificação como em regressão.

Em 1965, Vladimir Vapnik propôs um algoritmo para encontrar um hiperplano separador em uma classificação linear. Entretanto, em 1992, Bernhard Boser, Isabelle Guyon e Vapnik criaram uma classificação não linear, aplicando as Funções *Kernel*. Em 1995, Corinna Cortes e Vapnik sugeriram uma modificação no algoritmo, introduzindo variáveis de folgas para obter uma margem suave no hiperplano separador (CORTES e VAPNIK, 1995).

O SVM minimiza a probabilidade de classificação errada dos padrões.

O nome *Support Vector Machine* enfatiza a importância que os vetores mais próximos da margem de separação representam, uma vez que eles determinam à complexidade do SVM (CARVALHO, 2005).

Se comparada com Redes Neurais, o SVM pode ser considerado uma rede de apenas uma camada escondida, como mostra a figura 10:

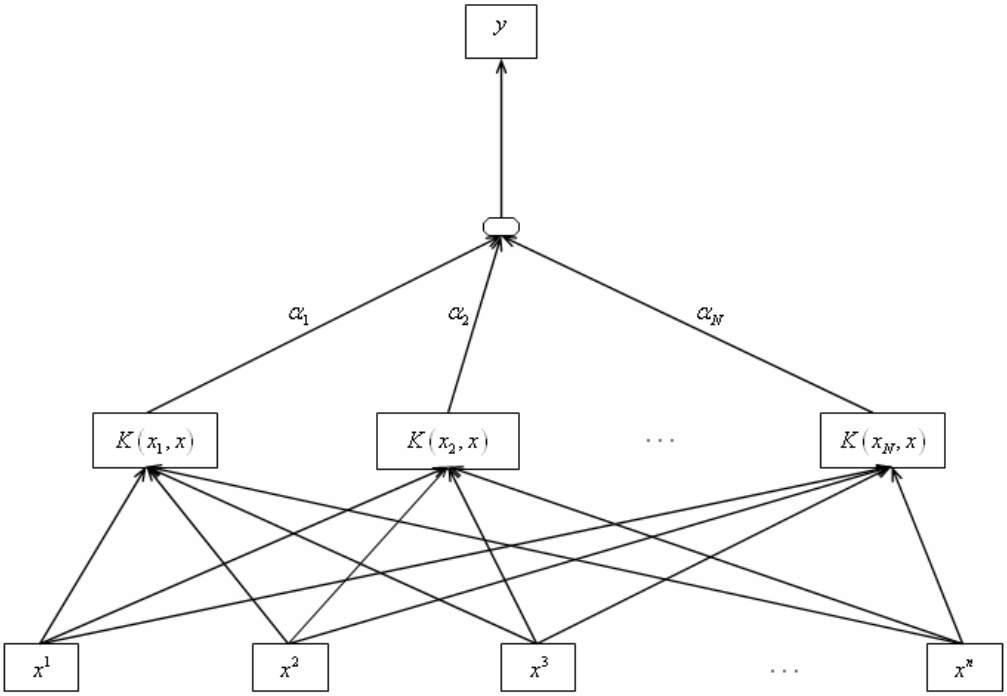

FIGURA 10 – REPRESENTAÇÃO DO SVM NA ESTRUTURA DE RNA
FONTE: Vapnik (1999)

Seu processo de aprendizagem é supervisionado, ou seja, os dados de treinamento, juntamente com suas respectivas saídas são apresentados à máquina para ajuste dos parâmetros.

O treinamento é realizado através da resolução de um problema quadrático convexo e assim, todo mínimo local é mínimo global.

A formulação teórica do SVM é apresentada no próximo capítulo, com mais detalhes, pois este é um dos objetivos desta dissertação.

2.5 PESQUISA OPERACIONAL

O problema do SVM está envolvido com a pesquisa operacional.

A pesquisa operacional é uma ciência que fornece ferramentas quantitativas no processo de análise de decisões. É uma técnica importante para determinar a melhor utilização de recursos e otimizar processos, sendo bastante empregada nas indústrias.

Entre o conjunto de métodos da pesquisa operacional encontram-se a programação linear e não-linear, esta envolve a programação quadrática, Teoria do Lagrangeado e Dualidade.

2.5.1 Programação Linear

Algumas situações reais podem ser descritas por uma função objetivo, a ser maximizada ou minimizada, satisfazendo restrições lineares de igualdade ou de desigualdade, constituindo assim um Problema Linear (MACHADO, 2001).

A forma padrão é representada por:

$$\begin{aligned} &Minimizar \quad f(x) \\ &s.a. \qquad\quad x \in D \end{aligned} \tag{2.9}$$

Onde $f(x)$ é uma função linear e D é o conjunto das restrições também lineares.

O Simplex é um método desenvolvido por George B. Dantzig (1947), criado para resolver problemas lineares de maneira bastante eficiente. Ocorreram melhoramentos neste método: como a Elaboração do Simplex Revisado[14], Teoria da Dualidade[15], Regras Lexicográficas[16] e outros métodos foram acrescidos para o melhoramento do simplex.

Para a aplicação do método simplex é necessário que o problema esteja na forma padrão, ou seja, as restrições devem ser de igualdade. Caso o problema não esteja na forma padrão é possível coloca-lo na forma padrão mediante transformação de variáveis e adição de variáveis de folga (MURTY, 1985).

As condições de otimalidade de um Problema Linear são as condições de Karush-Kuhn-Tucker (KKT). Tratando-se de um problema linear as condições de primeira ordem que garantem a convexidade são suficientes para garantir a existência de um ótimo global, as condições de segunda ordem tornam-se desnecessárias, pelo fato da Matriz Hessiana do Lagrangeano de problemas lineares ser nula.

[14] O simplex revisado foi proposto por Dantzig, Orchard-Hays e Wolfe (1953/1954).
[15] A Teoria da Dualidade juntamente com o algoritmo Dual Simplex foram propostos por Lemke (1954).
[16] Regras Lexicográficas para resolver problemas de degeneração e iterações cíclicas proposto por Beale (1955), Dantzig, Orden e Wolfe (1955).

2.5.2 Programação não linear

Uma generalização do problema não linear tem a seguinte forma:

$$\begin{aligned} &Minimizar \quad f(x) \\ &s.a. \qquad\quad x \in D \end{aligned} \tag{2.10}$$

Onde $f(x)$ pode ser uma função não linear e D o conjunto das restrições. Esse problema possui uma solução global garantida pelo teorema a seguir:

Teorema de Weierstrass: Sejam $D \subset \mathbb{R}^n$ um conjunto compacto não vazio e $f : D \to \mathbb{R}$ uma função contínua. Então o problema de minimizar a função $f(x)$ com $x \in D$ possui solução global.

A demonstração deste teorema encontra-se em Izmailov e Solodov (2005).

Os problemas não lineares não possuem um método que sobressaia aos outros, como em problemas lineares existe o algoritmo Simplex e o método de Karmarkar. Existem vários métodos para resolver um problema não linear, como por exemplo, Multiplicadores de Lagrange, Método de Newton, Método do Gradiente, Método Quase-Newton, Métodos de Filtro, entre outros. Cada método tem suas características, sobressaindo para determinados problemas e possuem diferentes tipos de convergência (FRIEDLANDER, 1994).

2.5.2.1 Programação quadrática

Problemas gerais de programação quadrática possuem uma função-objetivo quadrática e estão sujeitos a restrições lineares ou quadráticas (IZMAILOV e SOLODOV, 2005).

Uma classe mais específica de problemas quadráticos trata de problemas quadráticos convexos, na qual se encontra o problema do SVM.

Pode-se definir um problema quadrático primal da seguinte maneira:

$$\begin{aligned} &\textit{Minimizar} \quad f(w) \\ &s.a. \qquad \begin{aligned} &g_i(w) \le 0,\ i = 1,...,k \\ &h_i(w) = 0,\ i = 1,...,m \end{aligned} \end{aligned} \tag{2.11}$$

Onde $f(w)$ é a função objetivo, g_i, h_i são as funções restringindo o problema.

As definições e teoremas a seguir podem ser encontrados em Izmailov e Solodov (2005).

Um conjunto $D \subset \mathbb{R}^n$ é chamado convexo se para quaisquer $x \in D$, $y \in D$ e $a \in [0,1]$, tem-se que a combinação convexa $ax + (1-a)y \in D$ (IZMAILOV e SOLODOV, 2005).

Consequentemente, qualquer subespaço de $\mathbb{R}^n$, qualquer semi-espaço em $\mathbb{R}^n$ e qualquer hiperplano em $\mathbb{R}^n$ são conjuntos convexos.

Se $D \subset \mathbb{R}^n$ é um conjunto convexo, diz-se que a função $f : D \to \mathbb{R}$ é convexa em D quando para quaisquer $x \in D$, $y \in D$ e $a \in [0,1]$, tem-se:

$$f(ax + (1-a)y) \le af(x) + (1-a)f(y) \tag{2.12}$$

Como ilustra a figura 11:

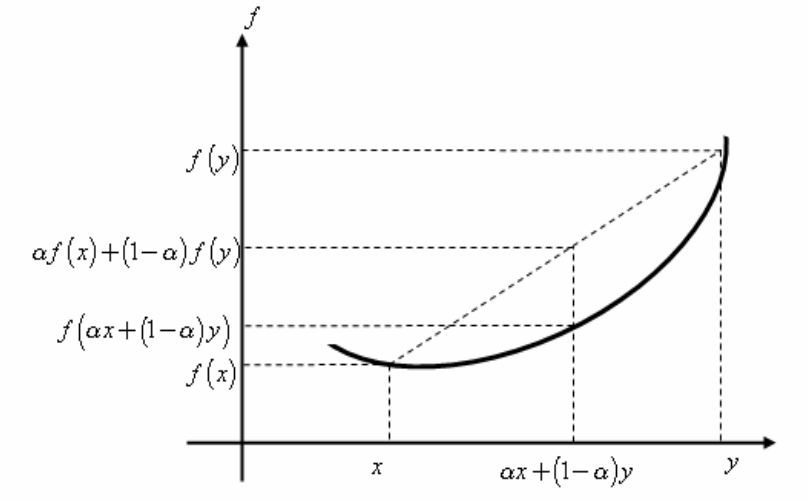

FIGURA 11 – ILUSTRAÇÃO DA FUNÇÃO CONVEXA
FONTE: Izmailov e Solodov (2005).

A vantagem de se ter uma função convexa em um problema de minimização é ter a certeza de que a mesma possui solução global, confirmada pelo teorema a seguir:

Teorema da Minimização Convexa: Sejam $D \subset \mathbb{R}^n$ um conjunto convexo e $f : D \to \mathbb{R}$ uma função convexa em D. Então todo minimizador local da função $f(x)$ em $x \in D$ é global.

O termo minimização de uma função convexa é equivalente a maximização de uma função côncava.

2.5.2.2 Condições de otimalidade de Kuhn-Tucher

O problema que se almeja abordar é o de reconhecer quando foi atingida a solução ótima, cuja função objetivo é não linear e existem restrições, neste caso não é suficiente analisar a derivada. As condições para otimalidade são: a primeira é que a função objetivo deve ser convexa para minimização ou côncava para maximização. A segunda condição é que as restrições do problema devem delimitar um conjunto convexo (EHRICH, 1980).

2.5.3 Teoria do Lagrangeano

Foi desenvolvida por Lagrange em 1797, e depois teve mais avanços com Kuhn e Tucker.

Essa teoria tem a finalidade de caracterizar a solução de um problema de otimização, inicialmente, quando não existem restrições de desigualdade.

Teorema de Fermat: Uma condição mínima para $w*$ ser um mínimo de $f(w)$, $f \in C^1$, é $\frac{\partial f(w*)}{\partial w} = 0$. Desde que f seja uma função convexa, essa é uma condição suficiente.

Ou seja, este teorema afirma que se uma função é convexa e de classe C^1, basta encontrar o ponto com derivada primeira igual à zero, que será encontrado o mínimo da função.

Precisamente, o lagrangeano é definido como a função objetivo somada com a combinação linear das restrições, onde os coeficientes da combinação linear são chamados de Multiplicadores de Lagrange (CRISTIANINI e SHAWE-TAYLOR, 2000).

Definição de Função Lagrangeana: Dado um problema de otimização com função objetivo $f(w)$, e restrição de igualdade $h_i(w)=0$, $i=1,...,m$, define-se a função lagrangeana como:

$$L(w,\alpha)=f(w)+\sum_{i=1}^{m}\alpha_i h_i(w) \tag{2.13}$$

Onde os coeficientes α_i são os Multiplicadores de Lagrange.

Em programação não linear, os Multiplicadores de Lagrange tem o mesmo significado que as variáveis duais na programação linear (EHRLICH, 1980).

Teorema de Lagrange: Uma condição necessária para um ponto w^* ser o mínimo da função $f(w)$ sujeito a $h_i(w)=0$, $i=1,...,m$ com $f, h_i \in C^1$, é:

$$\frac{\partial L(w^*,\alpha^*)}{\partial w}=0$$

$$\frac{\partial L(w^*,\alpha^*)}{\partial \alpha}=0 \tag{2.14}$$

As equações 2.14 devem ser válidas para algum valor de α^*.

As condições acima são suficientes desde que $L(w,\alpha^*)$ seja uma função convexa de w.

Definição de Função Lagrangeana Generalizada: Dado um problema de otimização do problema 2.11, com domínio $\Omega \subset R^n$, define-se a função Lagrangeana Generalizada como:

$$\mathcal{L}(w,\alpha,\beta)=f(w)+\sum_{i=1}^{k}\beta_i g_i(w)+\sum_{i=1}^{m}\alpha_i h_i(w)=f(w)+\beta' g(w)+\alpha' h(w) \tag{2.15}$$

Assim, o lagrangeano contempla restrições de igualdade e de desigualdade.

2.5.4 Dualidade

A teoria de dualidade se baseia em associar ao problema original (primal) um outro problema, denominado dual, que sob certas condições é equivalente ao primal e que, às vezes, é mais fácil de ser resolvido. As relações de dualidade são muito úteis na teoria e nas técnicas computacionais (ISMAILOV e SOLODOV, 2005).

A dualidade mais forte é obtida em problemas primais de minimização onde a função é convexa, o que se torna mais conveniente obter a função lagrangeana.

Dada à função Lagrangeana, pode-se definir o problema dual lagrangeano:

Definição de Problema Dual Lagrangeano: Dado um problema primal 2.11, com $\Omega \in D$, o problema Dual Lagrangeano é dado por:

$$\begin{aligned} &\textit{Maximizar} \quad \theta(\alpha,\beta) \\ &\textit{s.a.} \qquad\quad \alpha \geq 0 \end{aligned} \tag{2.16}$$

onde:

$$\theta(\alpha,\beta) = \inf_{w\in\Omega} L(w,\alpha,\beta)$$

Definição de *Gap* Dual: A diferença entre os valores da função objetivo do modelo primal e do modelo dual é conhecido como *gap* dual (CRISTIANINI e SHAWE-TAYLOR, 2000).

A informação do *gap* dual está relacionada com o fato de que se a solução ótima é encontrada quando o valor da função objetivo primal w é igual à função objetivo dual θ. Portanto, quanto mais próximo de zero estiver o *gap* dual, mais próximo está da solução ótima do problema, como afirma o teorema a seguir:

Teorema forte da dualidade: Dado um problema de otimização 2.11 com domínio convexo $\Omega \subset R^n$, onde g_i, h_i são funções afins, o *gap* dual é zero.

Teorema de Kuhn-Tucker: Dado um problema de otimização 2.11 com domínio convexo $\Omega \subset R^n$, com $f \in C^1$ e g_i, h_i funções afins, as condições necessárias e suficientes para que w^* seja ponto de ótimo é que existam α^* e β^* tais que:

$$\begin{aligned}
\frac{\partial L(w^*,\alpha^*,\beta^*)}{\partial w} &= 0 \\
\frac{\partial L(w^*,\alpha^*,\beta^*)}{\partial \alpha} &= 0 \\
\beta_i g_i(w^*) &= 0 \\
g_i(w^*) &\leq 0 \\
\beta_i^* &\geq 0\ ,\ i = 1,...,k\ .
\end{aligned} \tag{2.17}$$

O resultado acima mostra que a definição de Multiplicador de Lagrange no contexto de dualidade é uma extensão da noção do multiplicador nas condições de otimalidade de Karush-Kuhn-Tucker.

A importância desta fundamentação teórica para o trabalho está relacionada ao desenvolvimento da técnica do Support Vector Machine bem como suas variações, a qual depende de um problema de minimização quadrático convexo, que possui solução ótima.

Para encontrar uma solução pode-se utilizar o problema dual, mas esse depende da teoria do lagrangeano, que está relacionado às condições de Karush- Kuhn-Tucker.

No próximo capítulo são vistos o desenvolvimento do Support Vector Machine, suas formulações primal e dual.

3 SVM

A idéia inicial de criar algoritmos para reconhecer padrões surgiu através de Fisher em 1936. Mais tarde, em 1962, Rosenblatt sugeriu o algoritmo Perceptron de aprendizagem.

As técnicas conhecidas são capazes de resolver determinados conjuntos de problemas, visto que, a construção da técnica impõe alguns limites. Portanto, a busca por novos algoritmos intensifica-se.

Em 1992, após três décadas de pesquisa, foi fundamentada a técnica de reconhecimento usando vetores suportes, *Support Vector Machine*. O objetivo inicial era encontrar um hiperplano separador que tivesse a máxima margem, ou seja, a maior distância entre os vetores suportes, usando a metodologia de aprendizagem. Em 1963, Vladimir Vapnik propôs o algoritmo do hiperplano de máxima margem, uma classificação linear. As pesquisas continuaram com a intenção de encontrar uma separação não linear. Em 1992, Bernhard Boser, Isabelle Guyon e Vladimir Vapnik sugerem a separação não linear, utilizando as funções *kernel*, que transformam o espaço de entrada em um espaço de dimensão maior, onde os dados são linearmente separáveis.

Em 1995, Corinna Cortes e Vladimir Vapnik propõem uma modificação que permite dados classificados incorretamente, introduzindo variáveis de folga, o hiperplano de separação com margens flexíveis.

As buscas por métodos melhores não cessam, foram criadas algumas variações para o SVM, sendo ideais para certas condições sobre o conjunto de dados, por exemplo, número elevado de pontos.

3.1 METODOLOGIA DE APRENDIZAGEM

Um sistema de aprendizagem deve ter a capacidade de, após observar o comportamento de vários pares de entrada e saída $\left\{x^i, y_i\right\}$, $i = 1,...,l$, imitar o comportamento e gerar saídas próximas de y_i a partir de entradas próximas a x^i.

Quando o número de saídas ou classes for finito, essa classificação é denominada reconhecimento de padrões. Se existirem apenas duas saídas possíveis,

denomina-se classificação binária. E quando existirem infinitas saídas possíveis, então se denomina problema de regressão (CRISTIANINI e SHAWE-TAYLOR, 2000).

Esse processo de aprendizagem, treinamento da máquina, refere-se à fase de adquirir conhecimento, ou seja, reter informações relevantes de certa base de dados, para posteriormente utilizar.

No treinamento, o objetivo principal é ajustar os parâmetros livres do sistema, e assim encontrar uma ligação entre os pares de entrada e saída.

Um método de treinamento é o treinamento supervisionado, fornecendo a máquina exemplos de entrada, juntamente com as saídas desejadas de cada exemplo. Desta maneira, os acertos são valorizados e os erros penalizados, possibilitando que a máquina adquira aprendizado (Figura 12).

Também existe o treinamento não supervisionado, no qual não se tem conhecimento das saídas do conjunto. Para esse treinamento é realizada uma relação de agrupamento para encontrar características semelhantes e agrupar (SHAWE-TAYLOR e CRISTIANINI, 2004).

A utilização da máquina de aprendizagem possui duas fases: treinamento e aplicação. Na primeira a máquina utiliza um conjunto de dados específicos para adquirir conhecimento.

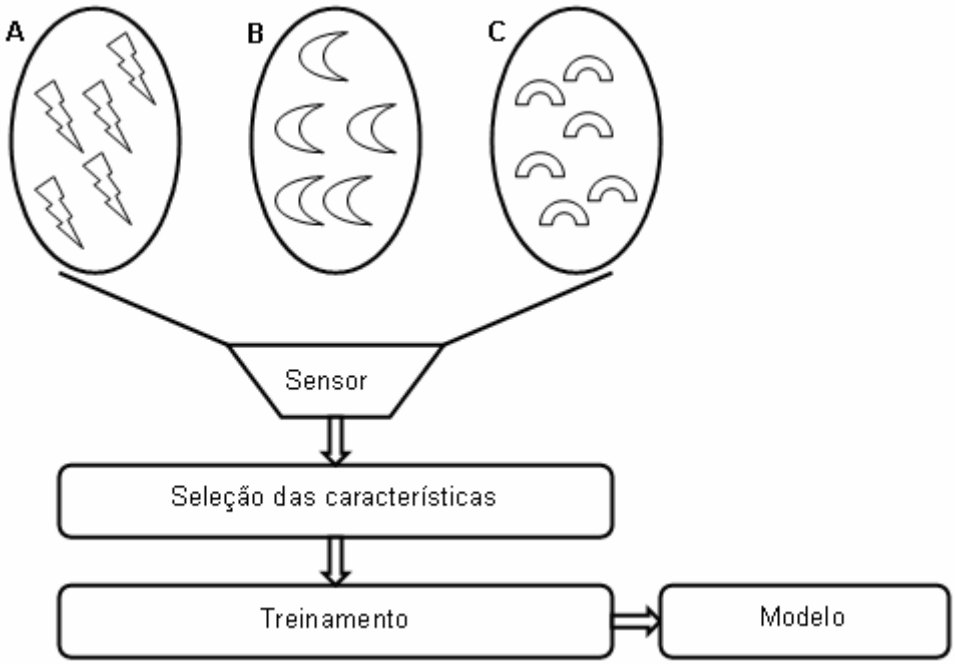

FIGURA 12 – ESQUEMA DE TREINAMENTO SUPERVISIONADO
FONTE: A autora (2008)

Na segunda é utilizado um conjunto de teste, no qual a máquina irá utilizar o conhecimento adquirido para fazer a classificação (Figura 13).

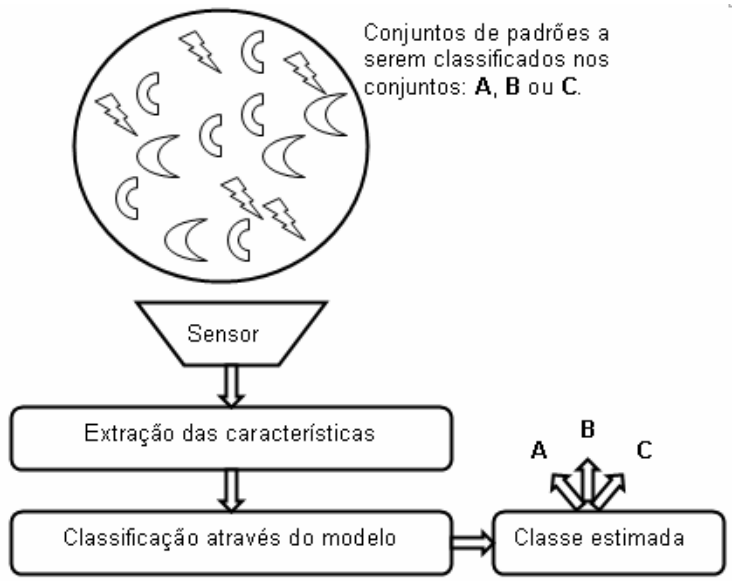

FIGURA 13 – ESQUEMA DE TESTE
FONTE: A autora (2008)

Para ocorrer sucesso é necessário que o conjunto de treinamento seja estatisticamente representativo, ou seja, represente os dados.

A metodologia de aprendizagem é muito atrativa, pois a taxa de aplicações potencialmente solucionadas por esta abordagem é muito grande.

Na prática, os problemas de aprendizagem podem manifestar dificuldades específicas. A primeira é que o algoritmo de aprendizagem pode apresentar ineficiência. A segunda é que o tamanho das hipóteses de saída pode se tornar muito grande e impraticável. A terceira é que se existir um número limitado de dados de treinamento muito específico a uma situação, esses dados podem aprender exageradamente e obter uma generalização muito pobre (*overfitting*). O quarto problema é que frequentemente os algoritmos de aprendizagem são controlados por um amplo número de parâmetros que são escolhidos por heurísticas tornando o sistema difícil e não confiável para utilização (CRISTIANINI e SHAWE-TAYLOR, 2000).

3.1.1 Modelo Primal

A forma original de modelar o problema do SVM é através do modelo primal que pode ser separável por um hiperplano ou por uma superfície. A seguir apresentam-se dois modelos de SVM, na forma dual, o primeiro que enfatiza a máxima margem de classificação, caso linear, e o segundo que considera os conjuntos de treinamento não lineares.

3.1.1.1 Modelagem primal com as margens rígidas

O SVM é uma metodologia que classifica os padrões em dois conjuntos, devido à metodologia utilizar margens de separação para os conjuntos, as quais são deslocamentos com distancia igual a um, denominando-se o conjunto de classe -1 e outro de classe +1.

Em virtude do SVM fazer uso da aprendizagem supervisionada, introduz-se uma notação para referenciar conjunto de treinamento, dados de entrada e saídas.

Seja o conjunto de treinamento:

$$S = \left\{ \left(x^1, y_1 \right), \left(x^2, y_2 \right), ..., \left(x^l, y_l \right) \right\} \subseteq \left(X \times Y \right)^l \tag{3.1}$$

onde:

l é o número de pontos no conjunto de treinamento

Cada ponto é representado por $\left\{ x^i, y_i \right\}$, para $i = 1, ..., l$, onde $x^i \in \mathbb{R}^n$ é o vetor de entrada, pertencente ao espaço de entrada[17], que representa a quantificação das características, e $y_i \in \left\{ -1, +1 \right\}$ é a saída binária correspondente. Dado um vetor de entrada x, a saída do SVM é representada pelo sinal da função $f\left(x \right)$.

O objetivo do modelo do SVM é encontrar um hiperplano ótimo com a maior margem de separação possível que separe os vetores da classe -1 da classe $+1$. Considerando-se, inicialmente, dois conjuntos linearmente separáveis, como por exemplo, ilustrado na figura 14:

---

[17] Espaço de entrada é o espaço vetorial no qual se representa cada vetor de entrada $x^i \in \mathbb{R}^l$

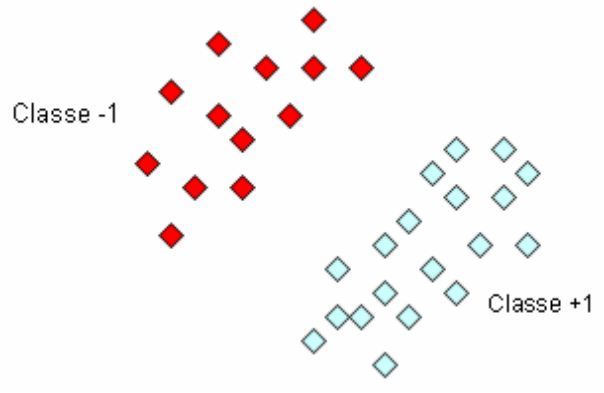

FIGURA 14 – ILUSTRAÇÃO DE CONJUNTOS LINEARMENTE SEPARÁVEIS
FONTE: A autora (2008)

Uma classificação linear pode ser obtida pela função real $f : X \subseteq \mathbb{R}^n \to \mathbb{R}$, da seguinte maneira: a entrada $x^i = \left(x_1^i, x_2^i, ..., x_l^i\right)$ é considerada de classe $+1$ se $f\left(x\right) \geq 0$ e caso contrário é considerada de classe -1 . Assim, a função decisão $f\left(x\right)$ é representada por:

$$f\left(x\right) = w^t x + b \tag{3.2}$$

onde:

$w \in \mathbb{R}^n$ é o vetor de pesos

$b \in \mathbb{R}$ é chamado de *bias*

Caso o espaço de características tenha dimensão dois, o hiperplano separador obtido é uma reta. Caso o espaço de características tenha dimensão n o hiperplano separador obtido é uma superfície de dimensão $n-1$. O hiperplano separador é uma superfície que divide o espaço de características em dois subespaços, sendo um para a classe -1 e o outro para a classe $+1$.

A classificação de cada padrão x do conjunto de treinamento é dada conforme à proximidade em relação às margens do hiperplano separador, ou seja, será classificado como pertencente a classe -1 se estiver mais próximo da margem negativa $w^t x + b = -1$ e será pertencente a classe $+1$ se estiver mais próximo da margem positiva $w^t x + b = 1$.

Um padrão x^i é considerado classificado corretamente se ele estiver fora da margem de separação de sua classe, ou seja, se $y_i = +1$ então deve satisfazer

$w^t x^i + b \geq 1$ e se $y_i = -1$ então deve satisfazer $w^t x^i + b \leq -1$. Como restrição do modelo do SVM considera-se a expressão:

$$y_i \left[w^t x^i + b \right] \geq 1 \ , \ i = 1, ..., l \tag{3.3}$$

Denomina-se margem funcional do ponto $\left(x^i, y_i \right)$ com relação ao hiperplano $f\left(x \right)$ a quantidade dada por:

$$\gamma_i = y_i \left(\left\langle w^t \cdot x^i \right\rangle + b \right) \tag{3.4}$$

Se $\gamma_i > 0$ então o ponto está classificado corretamente. A menor margem funcional é denominada margem funcional do hiperplano definida por γ, e, se estiver normalizada, denomina-se margem geométrica. Ou seja, se w é um vetor unitário, a margem geométrica é igual à margem funcional (CRISTIANINI e SHAWE-TAYLOR, 2000).

Fixado o valor $\gamma > 0$, pode-se definir a margem de folga ξ_i do ponto $\left(x^i, y_i \right)$ em relação ao hiperplano separador como:

$$\xi_i = \max \left\{ 0, \gamma - y_i \left(\left\langle w \cdot x^i \right\rangle + b \right) \right\} \tag{3.5}$$

A definição 3.5 informa a quantidade que o ponto falha para obter a margem funcional γ.

O objetivo é obter um hiperplano com a melhor generalização e robustez, dado que existem muitas possibilidades de separação dos conjuntos, como ilustrado na figura 15:

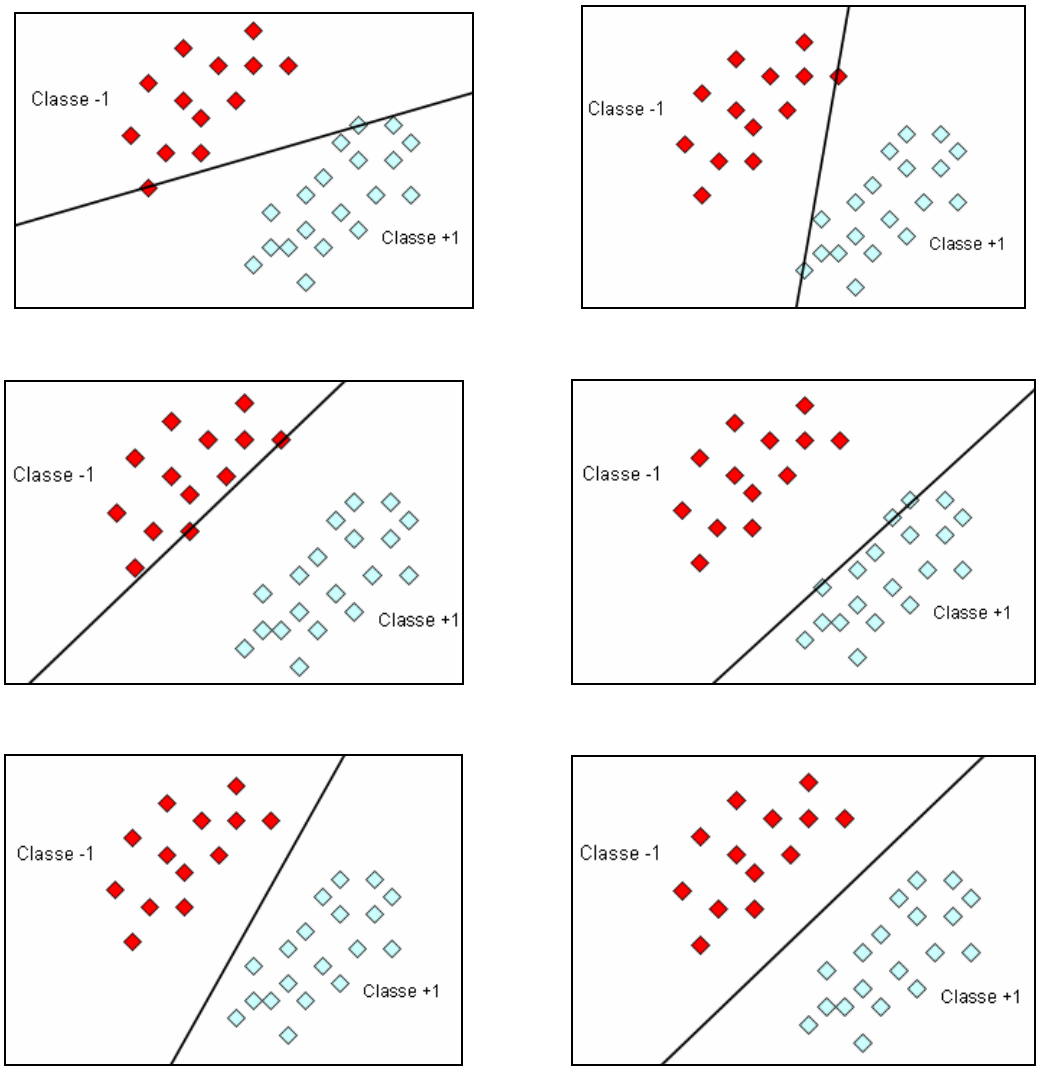

FIGURA 15 – POSSIBILIDADES DE HIPERPLANOS SEPARADORES
FONTE: A autora (2008)

O processo de treinamento do SVM consiste na obtenção de valores para os pesos w e do termo *bias* b de forma a maximizar a distancia entre as margens. Desta maneira, o SVM se torna robusto a pequenas variações no conjunto de treinamento, possibilitando uma melhor generalização, conforme a figura 16:

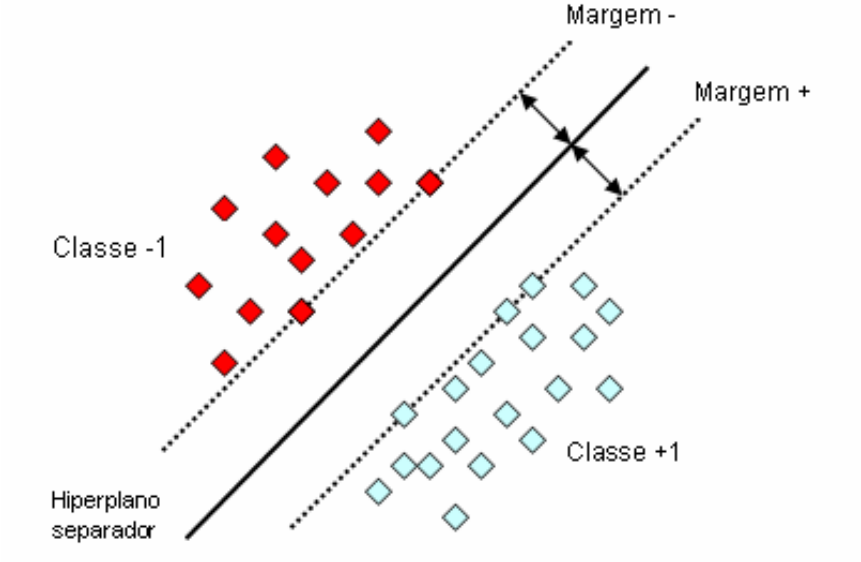

FIGURA 16 – SEPARAÇÃO ÓTIMA DE DOIS CONJUNTOS LINEARMENTE SEPARÁVEIS
FONTE: Carvalho (2005)

Considerando as margens do hiperplano de separação como $w^t x + b = +1$ e $w^t x + b = -1$, almeja-se obter a maior distância entre as margens para encontrar a melhor generalização e robustez do modelo do SVM. Conforme descrito pelo teorema a seguir:

Teorema: Caso os conjuntos sejam linearmente separáveis, tem-se que o hiperplano encontrado $w^t x + b = 0$ separa os dois conjuntos, cujas margens são definidas por:

$$\begin{aligned} w^t x + b &= +1 \\ w^t x + b &= -1 \end{aligned} \tag{3.6}$$

A distância entre as margens é dada por $d = \frac{2}{\|w\|}$.

Prova: Primeiramente, as margens são escritas como: $x^t w' + b' = \gamma$ a margem da classe positiva e $x^t w' + b' = -\gamma$, da classe negativa, onde γ é uma constante. Dividindo ambas as equações por γ e substituindo $\frac{w'}{\gamma}$ por w e $\frac{b'}{\gamma}$ por b obtem-se que as duas margens são $w^t x + b = 1$ e $w^t x + b = -1$.

O vetor w é um vetor perpendicular às margens $w^t x + b = 1$ e $w^t x + b = -1$, tomando βw o vetor que intercepta o hiperplano $w^t x + b = -1$ e αw o vetor que

intercepta o hiperplano $w^t x + b = 1$, a distância entre os hiperplanos pode ser medida por $d = \|\alpha w - \beta w\| = |\alpha - \beta| \|w\|$.

Substituindo no hiperplano $x^t = (\alpha w)^t$ e $x^t = (\beta w)^t$, obtem-se duas equações:

$$\begin{aligned} (\alpha w)^t w + b = 1 \quad &\Rightarrow \quad \alpha \|w\|^2 + b = 1 \\ (\beta w)^t w + b = -1 \quad &\Rightarrow \quad \beta \|w\|^2 + b = -1 \end{aligned} \tag{3.7}$$

Subtraindo a segunda equação da primeira obtemos: $(\alpha - \beta)\|w\|^2 = 2$

Portanto,

$$d = |\alpha - \beta| \|w\| = \frac{2}{\|w\|^2} \|w\| = \frac{2}{\|w\|} \tag{3.8}$$

□

Logo, para obter o melhor hiperplano, ou seja, aquele hiperplano que está mais longe dos dois conjuntos, deve-se maximizar $d = \frac{2}{\|w\|}$,ou de forma equivalente, minimizar $\frac{1}{2} w^t w$.

A maximização da margem de separação entre as classes, quando os conjuntos são linearmente separáveis, com o objetivo de obter um hiperplano ótimo, através do SVM, é resolvida através de um problema de otimização restrito.

O problema primal do SVM é dado por:

$$\begin{aligned} \text{Min} \quad & \frac{1}{2} w^t w \\ \text{s.a.} \quad & w^t x + b \geq 1, \; x \in +1 \\ & w^t x + b \leq 1, \; x \in -1 \end{aligned} \tag{3.9}$$

Ou, de forma compactada, dado por:

$$\begin{aligned} &\text{Min} \quad \frac{1}{2} w^t w \\ &\text{s.a.} \quad y_i \left[w^t x + b \right] \geq 1 \end{aligned} \tag{3.10}$$

onde:

w, b são variáveis

$w \in \mathbb{R}^n$ e $b \in \mathbb{R}$

Um problema de otimização como o problema primal do SVM, pode ser descrito de forma geral como: “Achar os valores de w que minimizem a função $f(w)$, sujeito às restrições $g_i(w) \leq 0$ e $h_i(w) = 0$ ”. Quando a função custo $f(w)$ é uma função quadrática em w, e suas restrições $g_i(w)$ e $h_i(w)$ são lineares, esse problema é denominado problema quadrático. Quando o problema quadrático possuir a função convexa então existe uma única solução global, ou seja, uma única solução ótima.

Portanto, existe um único conjunto de valores de w que torna a função custo $f(w)$ a menor possível.

Pesquisadores de redes neurais e estatística usam esse tipo simples de classificador, chamando-o respectivamente de *perceptron* e Discriminante Linear, onde o vetor w é referenciado como vetor de pesos e b conhecido por limiar ou *bias*.

3.1.1.2 Modelagem primal com margens flexíveis

São raros os casos reais em que os conjuntos são linearmente separáveis.

Seja o conjunto de treinamento S não separável linearmente. O problema primal apresentado anteriormente pode separar padrões incorretamente, desviando seu valor na função decisão.

Para impedir tal situação, e também aumentando a capacidade de generalização do SVM, acrescentam-se variáveis de folga, $\xi_i \geq 0$, associadas para cada vetor de treinamento x^i, sendo acrescida na restrição do SVM como $y_i \left[w^t x + b \right] \geq 1 - \xi_i$.

As variáveis de folga são obtidas durante o processo de treinamento do SVM.

Quando um padrão x^i está separado corretamente, tem-se $\xi_i = 0$. Caso um padrão x^i esteja entre a margem de separação de sua classe e o hiperplano separador, tem-se $0 < \xi_i < 1$. Se o padrão x^i estiver separado incorretamente, a variável de folga correspondente terá valor $\xi_i > 1$. Ou seja, sempre que um padrão não estiver dentro de sua margem, a variável de folga ξ_i é positiva, conforme ilustrado na figura 17:

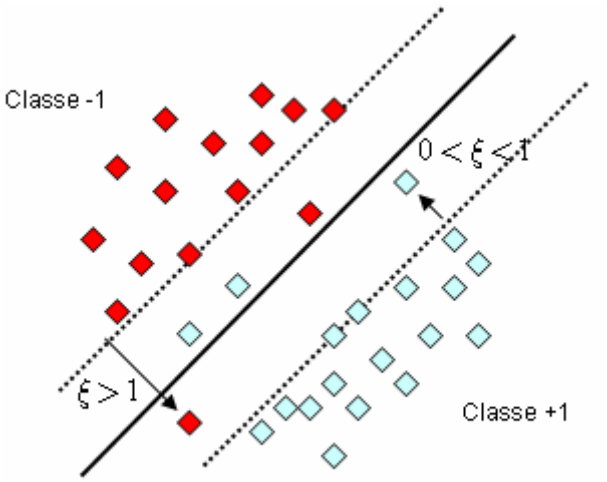

FIGURA 17 – SEPARAÇÃO RUIM DE DOIS CONJUNTOS NÃO SEPARÁVEIS LINEARMENTE
FONTE: Carvalho (2005)

As variáveis de folga indicam a distância de um padrão em relação à margem correspondente a sua classe. Ou seja:

$$\begin{aligned} w^t x + b &\geq 1 - \xi_i, \; x \in +1 \\ w^t x + b &\leq 1 - \xi_i, \; x \in -1 \end{aligned} \tag{3.11}$$

Dessa maneira, é possível expressar a classificação correta através de formulação compacta:

$$y_i \left[w^t x + b \right] \geq 1 - \xi_i \tag{3.12}$$

Este procedimento possibilita aceitar aqueles padrões que se situam fora da região de sua classe, impedindo que esses desviem o hiperplano separador $f(x)$. Esse hiperplano é denominado hiperplano de margem flexível (CORTES e VAPNIK, 1995).

Assim, o problema primal utilizando as variáveis de folga é dado por:

$$\begin{aligned} \text{Min} \quad & \frac{1}{2} w^t w + \mathrm{C} \cdot \sum_{i=1}^{l} \xi_i \\ \text{s.a.} \quad & w^t x^i + b \geq 1 - \xi_i, \; x \in +1 \\ & w^t x^i + b \leq 1 - \xi_i, \; x \in -1 \\ & \xi_i \geq 0 \qquad i = 1,...,l \end{aligned} \tag{3.13}$$

Ou, de maneira compactada, dado por:

$$\begin{aligned} \text{Min} \quad & \frac{1}{2} w^t w + \mathrm{C} \cdot \sum_{i=1}^{l} \xi_i \\ \text{s.a.} \quad & y_i \left[w^t x^i + b \right] \geq 1 - \xi_i \\ & \xi_i \geq 0 \qquad i = 1,...,l \end{aligned} \tag{3.14}$$

Onde w, b são variáveis, $w \in \mathbb{R}^n$, $b \in \mathbb{R}$ e C é um parâmetro que pondera os termos da minimização. O primeiro termo da função custo tem o objetivo de maximizar a margem, enquanto que o segundo termo, $\mathrm{C} \cdot \sum_{i=1}^{l} \xi_i$, minimiza o valor das variáveis de folga ξ_i, reduzindo o número de pontos que ficam do lado incorreto. Ou seja, o parâmetro C enfatiza maior ou menor importância das variáveis de folga, possibilitando que o modelo do SVM seja menos sensível à presença de pontos “mal comportados” no conjunto de treinamento.

3.1.2 Modelo Dual

Apresenta-se apenas a modelagem dual para o hiperplano com margem flexível de classificação.

Devido à natureza das restrições do problema primal, pode-se ter dificuldades na obtenção de sua solução. Utiliza-se a teoria do Lagrangeano para obter sua formulação dual, para o problema de otimização.

O problema dual possui a mesma solução do primal. Com isso, se a formulação do dual for mais simples, pode-se resolver o primal indiretamente, através da resolução do dual.

O problema Lagrangeano dual pode ser obtido acrescentando as restrições primais na função custo dual, por meio do uso de multiplicadores, denominados Multiplicadores de Lagrange. Os multiplicadores associados às restrições de desigualdade devem ser positivos e os associados à restrição de igualdade podem assumir quaisquer valores.

O problema Lagrangeano dual é conhecido como min-max, deve-se minimizar a nova função de custo em relação aos parâmetros primais e maximizá-la em relação aos parâmetros duais, tendo como única restrição uma expressão que garanta que os Multiplicadores de Lagrange da restrição de desigualdade sejam positivos, ou seja, achar o ponto de sela do problema Lagrangeano dual.

O problema primal 3.14 está na forma padrão. Aplicando a técnica do Lagrangeano, considerando o vetor $\vec{\alpha} = (\alpha_1, ..., \alpha_l)$ os l Multiplicadores de Lagrange, obtem-se a função a ser maximizada dada por:

$$\begin{aligned} &\text{Max } \mathrm{L}(w,b,\xi,\alpha) = W(w,b,\xi) - \sum_{i=1}^{l} \alpha_i \left[y_i \left(w^t \left(x^i \right) + b \right) - 1 + \xi_i \right] \\ &\text{s.a.} \qquad\qquad \alpha_i \geq 0 \end{aligned} \tag{3.15}$$

Sendo α_i o Multiplicador de Lagrange associado à i -ésima restrição de desigualdade do primal. Substituindo o valor de $W(w,b,\xi)$, obtém-se:

$$\begin{aligned} &\text{Max } \mathrm{L}(w,b,\xi,\alpha) = \frac{1}{2} w^t w + \mathrm{C} \cdot \sum_{i=1}^{l} \xi_i - \sum_{i=1}^{l} \alpha_i \left[y_i \left(w^t \left(x^i \right) + b \right) - 1 + \xi_i \right] \\ &\text{s.a.} \qquad\qquad \alpha_i \geq 0 \end{aligned} \tag{3.16}$$

Derivando o Lagrangeano em relação aos parâmetros primais, obtem-se:

$$\frac{\partial L(w,b,\xi,\alpha)}{\partial w} = w - \sum_{i=1}^{l} \alpha_i y_i \left(x^i\right)$$
$$\frac{\partial L(w,b,\xi,\alpha)}{\partial b} = -\sum_{i=1}^{l} \alpha_i y_i \tag{3.17}$$
$$\frac{\partial L(w,b,\xi,\alpha)}{\partial \xi} = C - \sum_{i=1}^{l} \alpha_i$$

Para minimizar o Lagrangeano dual em relação aos parâmetros primais, iguala-se a zero as derivadas parciais de primeira ordem em relação a cada um desses parâmetros:

$$w = \sum_{i=1}^{l} \alpha_i y_i \left(x^i\right)$$
$$\sum_{i=1}^{l} \alpha_i y_i = 0 \tag{3.18}$$
$$\sum_{i=1}^{l} \alpha_i = C$$

Substituem-se as expressões obtidas no próprio Lagrangeano dual, obtendo-se um problema de maximização dual, com restrição simples. Em virtude de obter apenas uma restrição e as variáveis limitadas o problema dual torna-se mais simples e mais fácil do que o problema primal.

$$\text{Max } \mathrm{L}(w,b,\xi,\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \left(x^i\right)\left(x^j\right)$$
$$\text{s.a.} \quad \sum_{i=1}^{l} \alpha_i y_i = 0 \tag{3.19}$$
$$0 \le \alpha_i \le C$$

O valor α_i é o Multiplicador de Lagrange correspondente ao padrão i e C é o parâmetro que limita o valor dos Multiplicadores de Lagrange no treinamento do SVM.

De acordo com as condições de KKT, as soluções ótimas α^*, (w^*, b^*) devem satisfazer:

$$\alpha_i^* \left[y_i \left(\left\langle w^*, x^i \right\rangle + b^* \right) - 1 + \xi_i \right] = 0 \,, \; i = 1,...,l \tag{3.20}$$

Isto implica que os α_i^* não zeros estão na margem funcional, chamados de vetores suportes e os demais são nulos (CRISTIANINI e SHAWE-TAYLOR, 2000).

O Multiplicador de Lagrange associado a cada ponto torna-se uma variável dual, ou seja, o conjunto de treinamento é importante para a solução final. Os pontos que não são vetores suportes não afetam o resultado (CRISTIANINI e SHAWE-TAYLOR, 2000).

Uma interessante conseqüência é que dado o conjunto de pontos de treinamento e supondo os valores ótimos dos Multiplicadores de Lagrange α^* encontrados pelo modelo dual do SVM, então o vetor de pesos $w = \sum_{i=1}^{l} y_i \alpha_i^* x^i$ classifica os dados com a margem máxima do hiperplano com margem geométrica dada por $\gamma = \frac{1}{\|w\|} = \left(\sum_{i=1}^{l} \alpha_i^* \right)^{-\frac{1}{2}}$.

3.1.3 Funções *Kernel*

Pode-se fazer um mapeamento dos dados não-lineares em um espaço de dimensão maior, onde os dados tornam-se linearmente separáveis (CARVALHO, 2005).

Os dados originais estão no espaço de entrada. Quando se faz uma função $\varphi : \mathbb{R}^n \to \mathbb{R}^p$, com $p > n$, os dados estão sendo mapeados no espaço *feature*[18].

O espaço *feature* consiste em um espaço no qual cada vetor de entrada é mapeado, por meio de $\varphi\left(x^i\right) \in \mathbb{R}^p$, em que p pode possuir qualquer valor maior do que n, dimensão do espaço de entrada, como ilustra a figura 18:

[18] Espaço *feature* é conhecido como espaço das características ou espaço dos atributos.

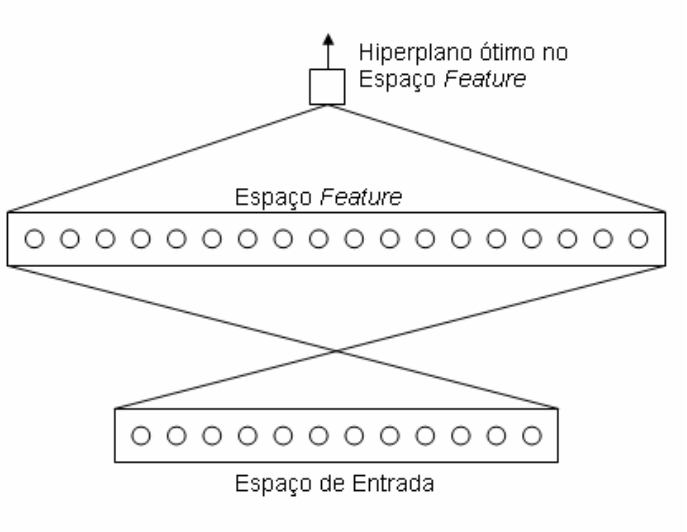

FIGURA 18 – MAPEAMENTO DOS DADOS DO ESPAÇO DE ENTRADA NO ESPAÇO *FEATURE*
FONTE: Vapnik (1998).

Para criar um espaço *feature*, por exemplo, usando uma função polinomial de grau dois, seja x um vetor no espaço de entrada com n coordenadas, podendo ser escrito como $x = \left(x^1, x^2, ..., x^n\right)$. O espaço *feature* teria em suas coordenadas da seguinte maneira:

- O espaço de entrada: $z^1 = x^1, ..., z^n = x^n$ (n coordenadas);
- O espaço de entrada ao quadrado: $z^{n+1} = \left(x^1\right)^2, ..., z^{2n} = \left(x^n\right)^2$ (n coordenadas);
- A multiplicação entre as dimensões do espaço de entrada $z^{2n+1} = x^1 x^2, ..., z^N = x^n x^{n-1}$ ($\frac{n(n+1)}{2}$ coordenadas).

Ou seja, um espaço de dimensão n seria levado num espaço *feature* de dimensão $N = \frac{n(n+3)}{2}$. Se o problema tem dimensão pequena, é possível construir o espaço *feature*. No entanto, se o problema tiver dimensão maior, por exemplo, $\mathbb{R}^{200}$, um polinomial de grau cinco geraria bilhões de coordenadas, tornando-se difícil de ser tratado computacionalmente.

Na função objetivo do problema dual do SVM existe um produto interno entre os pontos, o qual pode ser substituído por uma única função, denominada função *kernel*.

O *kernel* é uma função $K\left(x^i, x^j\right) = K\left(x^j, x^i\right) = \varphi\left(x^i\right) \cdot \varphi\left(x^j\right)$ para x^i e x^j pertencentes ao espaço de entrada, onde φ é um mapeamento do espaço de entrada no espaço das *features* (Figura 19).

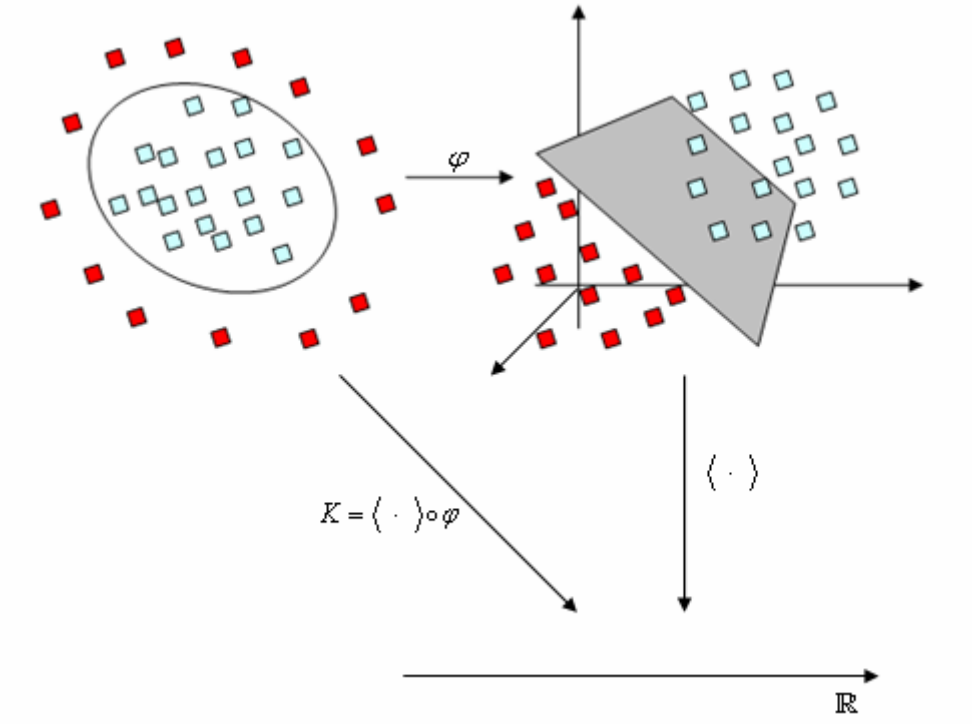

FIGURA 19 – MAPEAMENTO REALIZADO PELA FUNÇÃO *KERNEL* EM UM ESPAÇO DE DIMENSÃO MAIOR

FONTE: Carvalho (2005)

Os dados que estão no espaço de entrada não são separáveis por um hiperplano. A função φ faz uma transformação nos dados de maneira que, no espaço *features*, tornam-se separáveis.

Fazendo este produto interno sendo o usual, obtém-se uma superfície separadora linear, que separa apenas conjuntos separáveis. É possível obter uma superfície separadora, que separe conjuntos não separáveis linearmente, isso depende apenas da função *kernel* escolhida para cada conjunto de padrões a serem separados.

O hiperplano de separação é criado nos espaço *feature*, no qual os dados tendem a ser linearmente separáveis. No entanto, a utilização da superfície de decisão $f(x)=0$ ocorre no próprio espaço de entrada, de forma a separar os dados em classes diferentes. Essa superfície de separação pode gerar classes de forma interessante, no espaço de entrada, conforme pode ser visto na figura 20:

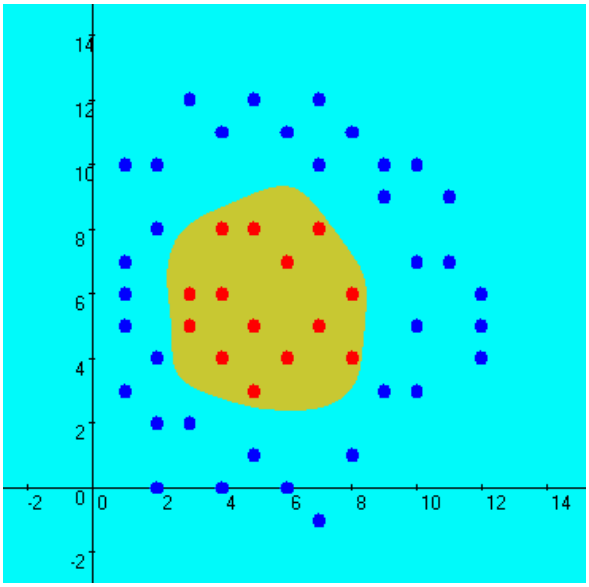

FIGURA 20 – SEPARAÇÃO DOS DADOS
FONTE: A autora (2008)

Os eixos x e y, que aparecem nas figuras, servem para localização dos pontos.

Na figura 20, os dados de uma classe -1 (pontos em azul) estão formando uma envoltória ao redor dos dados da classe $+1$ (pontos em vermelho), sendo impossível encontrar um hiperplano separador (nesse caso, uma reta) que separasse todos os pontos corretamente. A utilização da função *kernel*, nesse caso, o *kernel* gaussiano, obteve-se a separação ótima dos 49 dados, generalizando a situação dada, com uma margem de segurança. Nesse treinamento, com parâmetro $\sigma = 10$, obteve-se 18 vetores suportes.

Os dados apresentados na figura 21 possuem características diferenciadas, com as classes possuindo sub-regiões, contendo 615 pontos. A utilização de uma função *kernel* consegue encontrar uma superfície de separação, tendo o menor número de erros de classificação. Nesse exemplo, usou-se o *kernel* gaussiano com parâmetros $C = 1$ e $\sigma = 1$, obtendo-se 4,59% de erro. Para esta separação foram necessários 208 vetores suportes, sendo 134 vetores suportes *bounds*.

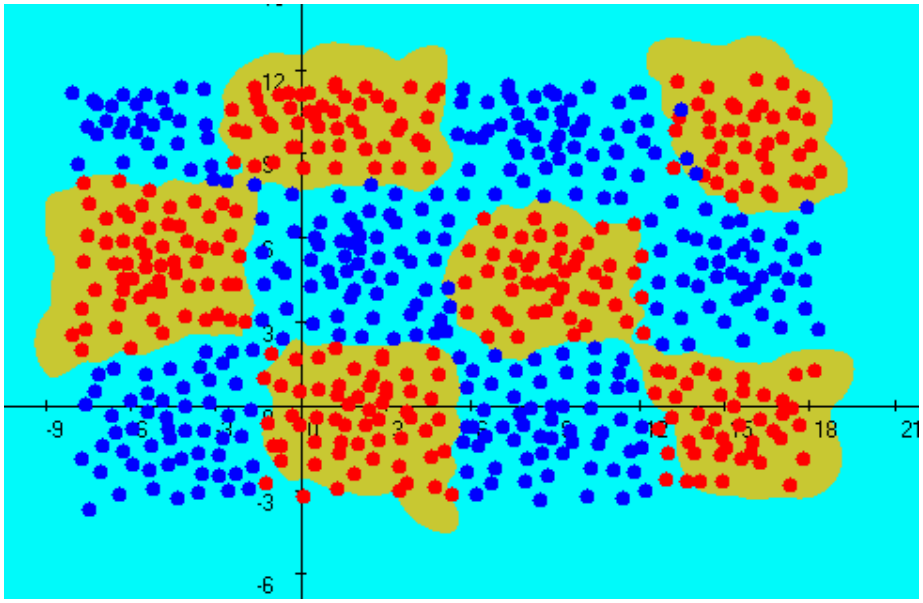

FIGURA 21 – SEPARAÇÃO DOS DADOS EM SUB-REGIOES
FONTE: A autora (2008)

A figura 22 possui 371 pontos, a separação realizada com a função *kernel* gaussiano de parâmetros $\sigma = 100$ e $C = 1$ resultou em 186 vetores suportes e apenas quatro pontos classificados incorretamente, gerando uma superfície separadora generalizada.

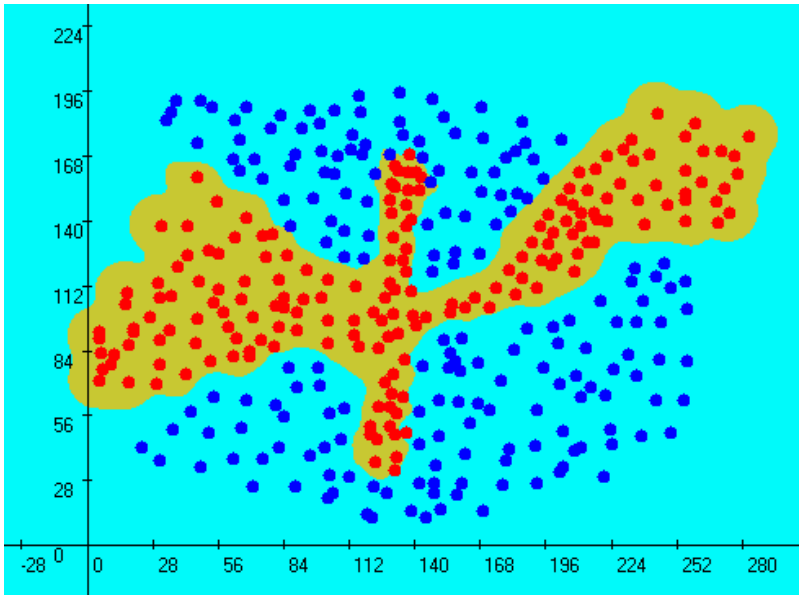

FIGURA 22 – SEPARAÇÃO DOS DADOS SEM REGIÕES DIFERENCIADAS
FONTE: A autora (2008)

Esse mapeamento tem o objetivo de facilitar a formulação de problemas complexos, que estão no espaço de entrada, através de superfícies lineares, do espaço *feature*.

Uma vez que o mapeamento do SVM é realizado por uma função *kernel*, e não diretamente pela função $\varphi(\cdot)$, nem sempre é possível saber exatamente qual mapeamento é efetivamente realizado, pois as funções *kernel* realizam um mapeamento implícito dos dados.

Se um hiperplano generaliza bem e teoricamente pode ser encontrado, resta o problema técnico de como tratar a alta dimensão do espaço *feature*. No entanto, para construção do hiperplano de separação ótimo no espaço feature, não é necessário considerá-lo de forma explicita.

Considerando a propriedade geral de produto interno no espaço de Hilbert, supõe-se um mapeamento φ do vetor $x \in \mathbb{R}^n$, $\varphi : \mathbb{R}^n \to \mathcal{H}$, dentro do espaço de Hilbert[19] com as seguintes coordenadas $\varphi_1(x), \varphi_2(x), ..., \varphi_n(x), ...$ De acordo com a Teoria de Hilbert-Schmidt, o produto interno no espaço de Hilbert tem representação equivalente a:

$$\langle \varphi_1 \cdot \varphi_2 \rangle = \sum_{p=1}^{\infty} \lambda_p \varphi_p(x^i) \cdot \varphi(x^j) \Leftrightarrow K(x^i, x^j) \tag{3.21}$$

Onde $K(x^i, x^j)$ é uma função simétrica satisfazendo a seguinte condição:

Teorema de Mercer: Para garantir que uma função simétrica contínua $K(x,z)$ em $L_2(C)$, C é um subconjunto compacto de $\mathbb{R}^n$, pode ser expandida como $K(x,z) = \sum_{p=1}^{\infty} \lambda_p \varphi_p(x) \cdot \varphi_p(z)$, com os coeficientes λ_p positivos, ou seja, $K(x,z)$ descreve um produto interno em algum espaço *feature*. A condição necessária e suficiente é:

$$\int_C \int_C K(x,z) g(x) g(z) dx dz \geq 0 \tag{3.22}$$

[19] Espaço de Hilbert $\mathcal{H}$ é um espaço vetorial de dimensão finita ou infinita.

A condição 3.22 deve ser válida a todas as funções $g \in L_2(C)$ (VAPNIK, 1998).

A demonstração desse teorema encontra-se em detalhes em Shawe-Taylor e Cristianini (2004).

A estrutura de produto interno no espaço de Hilbert que conduz a construção do SVM implica que para qualquer função *kernel* satisfazendo a condição de Mercer existe um espaço *feature* onde as funções geram um produto interno (VAPNIK, 1998).

A importância deste teorema está nas suas implicações, que asseguram a existência de um espaço *feature*. Uma das implicações é dada pela proposição a seguir:

Proposição: Seja C um espaço de entrada de dimensão finita com $K(x,z)$ uma função simétrica em C. Então $K(x,z)$ é uma função *kernel* se, e somente se, a matriz $\mathrm{K} = \left(K\left(x^i, x^j\right)\right)_{i,j=1}^{n}$ é semi-definida positiva, ou seja, os seus autovalores são não negativos.

A demonstração desse teorema encontra-se em Cristianini e Shawe-Taylor (2004).

Ou seja, uma função será *kernel* se for simétrica,

$$K(x,z) = \langle \phi(x) \cdot \phi(z) \rangle = \langle \phi(z) \cdot \phi(x) \rangle = K(z,x) \tag{3.23}$$

e satisfazer a desigualdade de Cauchy-Schwarz.

$$\begin{aligned} K(x,z)^2 &= \langle \phi(x) \cdot \phi(z) \rangle^2 \leq \|\phi(x)\|^2 \cdot \|\phi(z)\|^2 \\ &= \langle \phi(x) \cdot \phi(z) \rangle \langle \phi(x) \cdot \phi(z) \rangle \leq K(x,x) K(z,z) \end{aligned} \tag{3.24}$$

Segundo Nguyen et al (2006) não existe uma função *kernel* que seja superior a qualquer outra função. O desempenho depende do tipo de banco de dados de cada aplicação.

Algumas funções *kernel* mais utilizadas são:

1. Produto interno usual:

$$K\left(x^i, x^j\right) = \left(x^i\right)^t \cdot x^j \tag{3.25}$$

2. Polinomial Homogêneo:

$$K\left(x^i, x^j\right) = \left(\left(x^i\right)^t \cdot x^j\right)^p \tag{3.26}$$

onde:

p é o grau do polinômio

3. Polinomial Não Homogêneo:

$$K\left(x^i, x^j\right) = \left(\left(x^i\right)^t \cdot x^j + k\right)^p \tag{3.27}$$

onde:

p é o grau do polinômio

k é uma constante

4. Sigmoidal:

$$K\left(x^i, x^j\right) = \tanh\left(\kappa x^i \cdot x^j + k\right) \tag{3.28}$$

onde:

κ é um coeficiente

k é uma constante negativa

Hsu et al (2008) afirmam que este *kernel* não é válido para alguns parâmetros, em virtude de não ser um produto interno.

5. Gaussiano:

$$K\left(x^i, x^j\right) = e^{-\frac{\left\|x^i - x^j\right\|}{2\sigma^2}} \tag{3.29}$$

onde:

σ é um parâmetro

Essa função também é conhecida como RBF (*Radial Basis Function*[20]). Hsu et al (2008) afirmam que o *kernel* RBF é o mais adequado, possui números menores nos kernels, ou seja, $0 \le K\left(x^i, x^j\right) \le 1$, enquanto os demais *kernel* podem divergir, como por exemplo, o *kernel* polinomial.

Outras funções que satisfazem à condição de Mercer podem ser utilizadas como função *Kernel*, como, por exemplo, a função *kernel* ponderada, proposta por Nguyen et al (2006), a qual é da forma: $K_c = \sum_{i=1}^{m} \beta_i \times K_i$, onde $\beta_i \in [0,1]$, $\sum_{i=1}^{m} \beta_i = 1$ e $\left\{K_i / i = 1,...,m\right\}$ é o conjunto de funções *kernel* combinadas. Essa função *kernel* ponderado obteve resultados superiores às funções *kernel* polinomial e radial aplicadas em dados de pacientes, referentes à leucemia, câncer de cólon e câncer no pulmão, sendo utilizado Algoritmos Genéticos para determinar os coeficientes da função *kernel*.

Com as funções *kernel* o problema dual do SVM pode ser escrito como:

$$\begin{aligned} &\text{Max } \mathrm{L}(w,b,\xi,\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K\left(x^i, x^j\right) \\ &\text{s.a.} \qquad \sum_{i=1}^{l} \alpha_i y_i = 0 \\ &\qquad\qquad 0 \le \alpha_i \le C \end{aligned} \tag{3.30}$$

Tendo resolvido o problema dual, os valores ótimos α^* dos Multiplicadores de Lagrange são utilizados para encontrar o valor das variáveis primais, através das expressões:

$$\begin{aligned} w^* &= \sum_{i=1}^{l} \alpha_i^* y_i \varphi\left(x^i\right) \\ b^* &= \frac{1}{2}\left[w^{*t} x^+ + w^{*t} x^-\right] \end{aligned} \tag{3.31}$$

Onde x^+ é um padrão da classe $+1$ e x^- é um padrão da classe -1.

A saída obtida pelo SVM é da forma:

$$f(x) = \operatorname{sgn}\left[\sum_{i=1}^{l} \alpha_i^* y_i K\left(x^i, x\right) + b^*\right] \tag{3.32}$$

---

[20] Função de base Radial é uma função real que mede a distância da origem $\phi(x) = \phi(\|x\|)$ ou a distância até um ponto p, denominado centro, dado por $\phi(x, p) = \phi(\|x - p\|)$. Essa função é utilizada para construção de funções de aproximações.

Sendo $K\left(x^i, x\right)$ a função *kernel* escolhida. Portanto para reconhecer a qual conjunto um novo padrão pertence é possível ser verificado apenas pelo sinal da função.

Os valores nulos dos Multiplicadores de Lagrange não influenciam na obtenção dos pesos. Os padrões que tem o respectivo Multiplicador de Lagrange α não nulo são chamados de vetores suportes. Esses vetores são os únicos exemplos que contribuem para construção do hiperplano separador.

Segundo Cristianini e Shawe-Taylor (2000) é esperado o menor número de vetores suportes para obter-se uma melhor generalização.

Uma característica interessante do SVM é que apresenta esparsidade dos Multiplicadores de Lagrange ótimos, ou seja, após o treinamento do SVM vários Multiplicadores de Lagrange são nulos.

Para classificar os padrões do conjunto de treinamento como vetores suporte verificam-se os valores dos respectivos Multiplicadores de Lagrange, sendo:

Se $\alpha_i = 0,\ y_i \cdot f\left(x^i\right) > 1$ então x^i é considerado um vetor comum, que se situa do lado correto, na região da sua classe, e não influencia na construção do hiperplano separador ótimo.

Se $0 < \alpha_i < C,\ y_i \cdot f\left(x^i\right) = 1$ então x^i é um vetor suporte, situa-se sobre a margem da região da sua classe. É conhecido como vetor suporte *non-bound* (VS-NB).

Enfim, se $\alpha_i = C,\ y_i \cdot f\left(x^i\right) < 1$ então x^i é um vetor suporte *bound* (VS-bound). Ele pode se localizar entre a margem e o hiperplano separador, caso $0 < \xi_i < 1$. Ele pode estar na própria superfície de separação, caso $\xi_i = 1$, ou pode estar na região de classe oposta a sua, do outro lado da superfície de separação, caso $\xi_i > 1$ (CRISTIANINI e SHAWE-TAYLOR, 2000).

O número de graus de liberdade do SVM depende da função *kernel* escolhida. Algum conhecimento, “a priori”, pode ajudar no ajuste dos parâmetros da função *kernel* escolhida. Segundo Cristianini e Shawe-Taylor (2000), independente da classe de *kernel* e dos dados, é sempre possível encontrar os parâmetros do *kernel* que torna os dados separáveis. Lembrando que, forçar a separação de dados pode facilmente guiar a um particular *overfitting*, quando os dados possuem ruídos.

3.1.4 Pré-processamento dos dados

Nem sempre é adequado utilizar os dados da forma que são coletados, como por exemplo, imagens. Primeiramente, os dados das imagens devem ser convertidos em dados numéricos e transferidos em vetores, onde cada posição representa uma característica.

Outras formas de pré-processamento envolvem normalização e redução de dimensionalidade.

3.1.4.1 Normalização

O processo de normalização[21] consiste em atribuir uma escala aos dados, por exemplo, o valor dos dados tomados apenas no intervalo $[-1,+1]$ (HSU et al, 2008).

Esse processo deve ser feito quando os elementos de uma matriz de padrões estão com grandezas muito diferentes, o que pode tornar a matriz mal condicionada, possibilitando erros (BURDEN e FAIRES, 2003).

Uma importante vantagem desse processo é evitar números com escalas grandes dominando aqueles números em escalas muito pequenas e facilitar os cálculos durante o processo do SVM, por exemplo, cálculos de *kernel*, os quais envolvem produto interno.

3.1.4.2 Redução de dimensionalidade

A redução de dimensionalidade pode ser efetuada com três objetivos distintos: reduzir ruídos, extrair informações redundantes e facilitar a interpretação (GROBE, 2005).

O mais importante objetivo é evitar a chamada “maldição da dimensionalidade”. Esse problema ocorre quando se tem muitas variáveis, pois a função kernel gera uma dimensionalidade maior, acarretando num tempo computacional muito grande.

---

[21] Normalização é conhecida pelo termo *Scaling*

Esse processo é interessante para os casos onde os padrões são complexos e a dimensão dos mesmos é muito alta, como, por exemplo, em imagens, tornando inviável computacionalmente o reconhecimento de padrões.

Um processo bastante conhecido para reduzir a dimensão dos vetores é a Análise de Componentes Principais.

A Análise de Componentes Principais é uma técnica estatística que busca uma representação em dimensão menor de variáveis não correlacionadas. O objetivo é obter maior representatividade com respeito à matriz de variância-covariância Σ e os pares de autovalores e autovetores $(\lambda_1, e_1), (\lambda_2, e_2), ..., (\lambda_p, e_p)$ obtidos da mesma. A proporção da variância total explicada pela i-ésima variável é dada por:

$$\frac{\lambda_i}{\lambda_1 + \lambda_2 + ... + \lambda_p}, \; i = 1,2,..., p \tag{3.33}$$

Não se sabe o número de variáveis ideais para cada problema, então, na prática, são eliminadas as variáveis cujos autovalores são menores que certa fração (JOHNSON e WICHERN, 1998).

Esse processo é linear e, portanto, para dados não lineares é recomendável utilizar técnicas como Analise de Componentes Principais Não Lineares.

3.1.5 Parâmetros do SVM

Existem alguns parâmetros a serem estimados quando se utiliza a técnica de reconhecimento de padrões em SVM. Um desses parâmetros, encontrado na formulação dos modelos primal e dual é o C. Os demais parâmetros se encontram nas funções *kernel*.

De antemão, os valores dos parâmetros não são conhecidos. Não se sabe quais os melhores valores desses parâmetros para obter a melhor classificação dos dados.

É comum o procedimento de separar os dados de treinamento em dois grupos, um para o treinamento e o outro como desconhecido. Testar valores dos parâmetros no conjunto de treinamento e posteriormente, verificar a precisão destes valores no conjunto desconhecido.

Porém, uma versão melhor para obter os valores dos parâmetros é o *cross-validation* (validação-cruzada). A validação-cruzada υ *-fold* consiste na divisão do conjunto de treinamento em υ subconjuntos. Sequencialmente, um desses subconjuntos é testado usando o classificador treinado com os demais subconjuntos. Assim, a exatidão da validação-cruzada é dada pela porcentagem de dados classificados corretamente (HSU et al, 2008).

O procedimento da validação-cruzada evita problemas de *overfitting, e*xagero na classificação, provocando problemas na generalização. O *overfitting* ocorre devido a escolha dos parâmetros C e dos parâmetros necessários para o cálculo do *Kernel*.

A seguir, ilustrações das escolhas do parâmetro C. Na figura 23, os dados foram treinados de maneira exagerada, o SVM não aceita erros de classificação e a margem não foi maximizada. Em virtude disso, no teste, figura 24, os pontos ficaram classificados erroneamente, pois o hiperplano não conseguiu generalizar o problema.

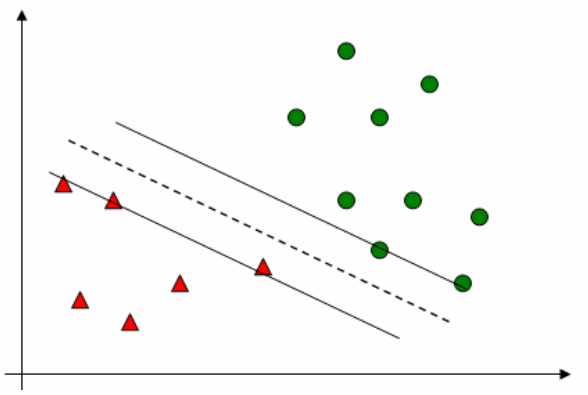

FIGURA 23 – TREINAMENTO EXAGERADO DOS DADOS
FONTE: Hsu et al (2008)

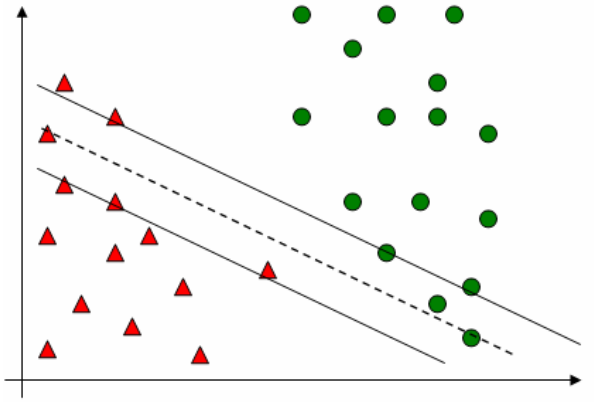

FIGURA 24 – ERROS NO TESTE
FONTE: Hsu et al (2008)

No entanto, se o parâmetro for ajustado corretamente isso não ocorre, como pode ser visto nas figuras 25 e 26.

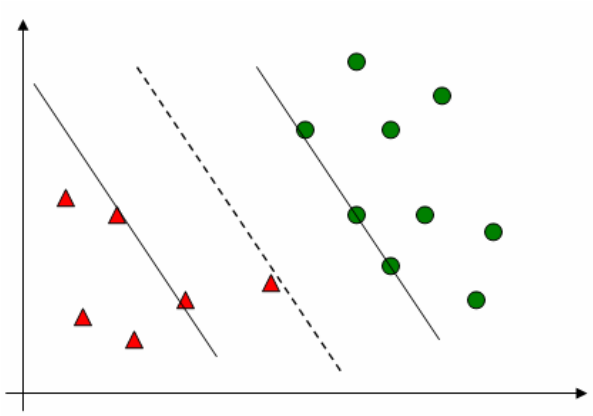

FIGURA 25 – TREINAMENTO MAXIMIZANDO A MARGEM ENTRE OS CONJUNTOS
FONTE: Hsu et al (2008)

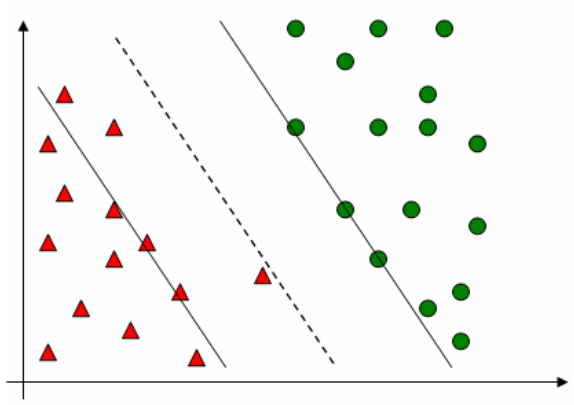

FIGURA 26 – TESTE DOS DADOS SEM ERROS DE CLASSIFICAÇÃO
FONTE: Hsu et al (2008)

O fato de alguns pontos estarem classificados incorretamente pode ser resultado de erros nos dados, ou seja, a classe ou a localização do ponto podem estar incorretas, ou como é o caso do ponto muito próximo do hiperplano separador.

O SVM tem o objetivo de separar ao máximo os conjuntos. Dependendo do valor do parâmetro C o treinamento do conjunto de dados dá mais importância à distância entre as margens (como visto na figura 25) ou aos pontos classificados incorretamente (figura 23).

O problema em encontrar um hiperplano que melhor generalize a separação dos dados está em definir o valor do parâmetro C. Hsu et al (2008) propuseram uma grade de busca (*Grid-search*) para encontrar os melhores valores do parâmetro C.

Estabeleceu-se tentar seqüências com crescimento exponencial, por exemplo, $C = 2^{-5}, 2^{-3}, ..., 2^{13}, 2^{15}$. Esse processo faz uma busca exaustiva do parâmetro.

Pode ocorrer *overfitting* na escolha dos parâmetros do *Kernel* como pode ser visto na figura 27. Recomenda-se fazer uma grade de busca grossa, identificar a melhor região e afinar a grade de busca sobre essa região.

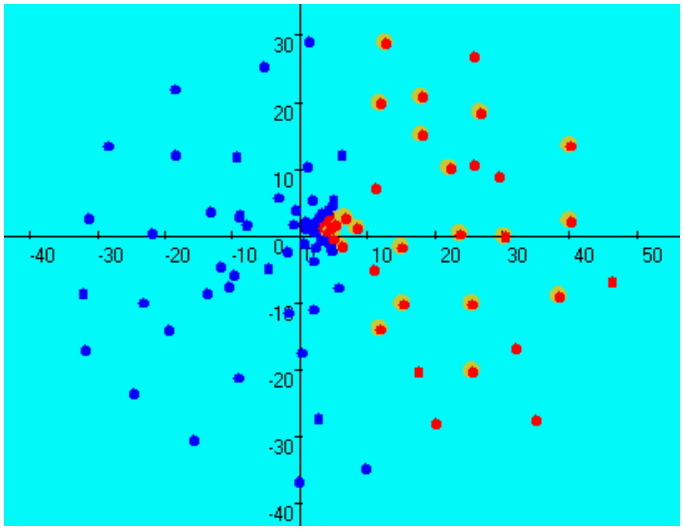

FIGURA 27 – TREINAMENTO EXAGERADO DOS DADOS
FONTE: A autora (2008)

Na figura 27, a região azul discrimina a classe -1 e a região amarela a classe $+1$. No treinamento realizado com 92 pontos, utilizou-se a função *kernel* gaussiano, com parâmetros $\sigma = 1$ e $C = 1$, obteve-se apenas um erro na classificação. No entanto, esse treinamento foi péssimo, não houve generalização, qualquer dado de teste com coordenadas deslocadas no plano pode ser considerado errado.

Para contornar esse problema de *overfitting*, treinou-se os dados com *kernel* gaussiano, cujos parâmetros foram $\sigma = 20$ e $C = 1$, obtendo-se doze pontos classificados incorretamente. Neste caso, verificou-se a separação foi generalizada, como mostra a figura 28:

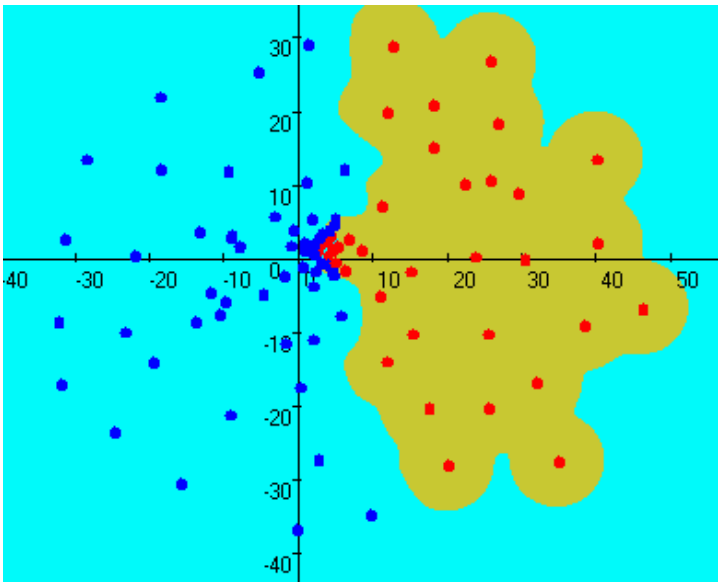

FIGURA 28 – TREINAMENTO COM MELHOR GENERALIZAÇÃO
FONTE: A autora (2008)

Na figura 28, o treinamento foi melhor, porém, ainda não está generalizado da melhor forma possível. Alternado a função kernel para polinomial não homogêneo de parâmetros $p = 3$ e $k = 1$, obtem-se o problema generalizado, contendo três pontos classificados incorretamente, como pode ser visto na figura 29:

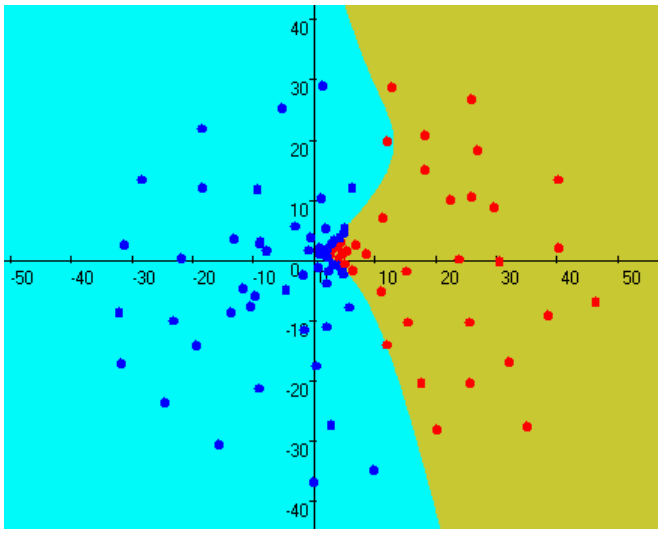

FIGURA 29 – TREINAMENTO GENERALIZADO
FONTE: A autora (2008)

Pode ser concluído, com essas ilustrações, que a escolha da função *kernel*, do valor dos parâmetros C e os específicos do *kernel*, como σ, κ, p e k, influenciam na

classificação, podendo levar a erros na classificação dos pontos de teste, por não haver generalização no treinamento dos dados, como exemplificado na figura 27.

Verifica-se que nas figuras 28 e 29, existem erros no treinamento dos dados. Isso ocorre pelo fato dos dados formarem um bico na fronteira das regiões, como pode ser visualizado na figura 30:

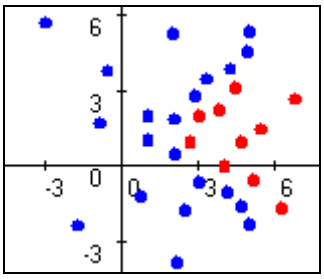

FIGURA 30 – VISUALIZAÇÃO DOS DADOS NA REGIÃO DE ERROS
FONTE: A autora (2008)

3.2 ALGUMAS CONSIDERAÇÕES

3.2.1 Número de padrões e número de características

O problema de reconhecimento de padrões pode ser visto como um problema matricial $X \cdot \Re = Y$, onde a matriz X representa a matriz dos dados, $\Re$ é o processo de reconhecimento e Y representa a saída, ou seja, as classes.

Dentro desse contexto, o reconhecimento consiste em encontrar um modelo que ligue os padrões às respectivas saídas. Portanto, se o número de padrões for muito pequeno e o número de características for elevado, o sistema pode ter diferentes representações, podendo se ajustar muito bem aos dados disponíveis, mas não encontrando o melhor ajuste a outros dados da aplicação.

Para evitar insucesso nas classificações o conjunto de treinamento, além de não ser um caso particular de alguns dados, deve ser representativo em relação a suas características.

3.2.2 Número de padrões e de vetores suporte

O número de vetores suporte (VS) não é definido, pois depende da dimensão do espaço onde ocorre a separação linear dos dados, ou seja, no espaço *feature*. Analisaremos um caso particular, onde os dados são linearmente separáveis, mas todos enfileirados, utilizando o produto interno usual com o parâmetro $C = 0.01$, como mostra a figura 31:

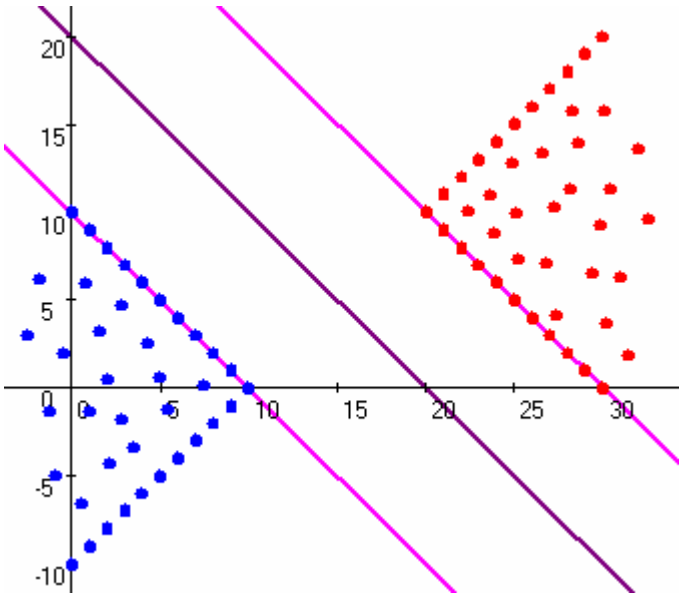

FIGURA 31 – HIPERPLANO LINEAR DE MARGEM MÁXIMA
FONTE: A autora (2008)

Intuitivamente, pelo fato de os conjuntos serem linearmente separáveis no $\mathbb{R}^2$, seria necessário apenas dois a três vetores suporte, mas não é o que ocorre. Na otimização do problema do SVM obtiveram-se 22 vetores suporte, sendo 2 vetores suporte *bounds* e 9 vetores suportes degenerados em cada um dos conjuntos, ou seja, $\alpha = 0$. Em virtude disso, nem sempre o menor número de vetores suporte são os que generalizam melhor a separação dos dados.

O espaço *feature* tem dimensão maior do que a dimensão do espaço de entrada. Não se tem conhecimento da dimensão que a função *kernel* gera no espaço *feature*, sendo impossível determinar o número mínimo de vetores suporte para o reconhecimento dos padrões.

Treinamentos realizados por Platt (1998), no banco de dados *Adult*, com valores binários de entrada 0 e 1, utilizando o produto interno e o parâmetro $C = 0.05$,

parâmetro escolhido através da validação cruzada, o número de vetores pode ser observado na tabela 1:

TABELA 1 – NÚMERO DE VS UTILIZANDO PRODUTO INTERNO USUAL NOS DADOS *ADULT*

DADOS	N.º PONTOS TREINADOS	N.º VS $0 < \alpha < C$	N.º VS *BOUNDS*($\alpha = C$)	TOTAL VS	% VS
A1A	1.605	42	633	675	42,06%
A2A	2.265	47	930	977	43,13%
A3A	3.185	57	1.210	1.267	39,78%
A4A	4.781	63	1.791	1.854	38,78%
A5A	6.414	61	2.370	2.431	37,90%
A6A	11.221	79	4.079	4.158	37,06%
A6A	16.101	67	5.854	5.921	36,77%
A7A	22.697	88	8.209	8.297	36,56%
A8A	32.562	149	11.558	11.707	35,95%

FONTE: Platt (1998)

Utilizando o *kernel* gaussiano com variância 10, $\sigma^2 = 10$, e $C = 1$, parâmetros escolhido através da validação cruzada, o número de vetores pode ser observado na tabela 2:

TABELA 2 – NÚMERO DE VS USANDO KERNEL GAUSSIANO NOS DADOS *ADULT*

DADOS	Nº PONTOS TREINADOS	Nº VS $0 < \alpha < C$	Nº VS *BOUNDS* ($\alpha = C$)	TOTAL VS	% VS
A1A	1.605	106	585	691	43,05%
A2A	2.265	165	845	1010	44,59%
A3A	3.185	181	1.115	1.296	40,69%
A4A	4.781	238	1.650	1.888	39,49%
A5A	6.414	298	2.181	2.479	38,65%
A6A	11.221	460	3.746	4.206	37,48%
A6A	16.101	567	5.371	5.938	36,88%
A7A	22.697	813	7.526	8.339	36,74%
A8A	32.562	1.011	10.663	11.674	35,85%

FONTE: Platt (1998)

Treinando outro banco de dados W*eb Pages* com 49.479 dados, contendo 300 características, utilizando o produto interno e o parâmetro $C = 1$, observa-se na tabela 3 um percentual de vetores suportes bem menor do que nos dados *Adult*:

TABELA 3 – NÚMERO DE VS USANDO PRODUTO INTERNO NOS DADOS *WEB PAGES*

DADOS	Nº PONTOS TREINADOS	Nº VS $0 < \alpha < C$	Nº VS *BOUNDS*($\alpha = C$)	TOTAL VS	% VS
W1A	2.477	123	47	170	6,86%
W2A	3.470	147	72	219	6,31%
W3A	4.912	169	107	276	5,62%
W4A	7.366	194	166	360	4,89%
W5A	9.888	214	245	459	4,64%
W6A	17.188	252	480	732	4,26%
W6A	24.692	273	698	971	3,93%

FONTE: Platt (1998)

A mesma observação pode ser concluída quando o kernel utilizado é o Gaussiano com variância $\delta^2 = 10$ e $C = 5$, conforme analisado na tabela 4:

TABELA 4 – NÚMERO DE VS USANDO PRODUTO INTERNO NOS DADOS *WEB PAGES*

DADOS	Nº PONTOS TREINADOS	Nº VS $0 < \alpha < C$	Nº VS *BOUNDS*($\alpha = C$)	TOTAL VS	% VS
W1A	2.477	439	43	482	19,46%
W2A	3.470	544	66	610	17,58%
W3A	4.912	616	90	706	14,37%
W4A	7.366	914	125	1.039	14,11%
W5A	9.888	1118	172	1.290	13,05%
W6A	17.188	1780	316	2.096	12,19%
W6A	24.692	2300	419	2.719	11,01%

FONTE: Platt (1998)

Platt criou um banco com dados artificiais para verificar o comportamento do SVM, em especial, analisando o comportamento do SVM em dados esparsos. Os dados foram criados aleatoriamente, com 300 características, sendo apenas 10% dos dados com valores 1, o restante nulos. Utilizando o produto interno e $C = 100$, que pode ser visto na tabela 5:

TABELA 5 – NÚMERO DE VS USANDO PRODUTO INTERNO NOS DADOS ARTIFICIAIS

Nº PONTOS TREINADOS	Nº VS $0 < \alpha < C$	Nº VS *BOUNDS*($\alpha = C$)	TOTAL VS	% VS
1.000	275	0	275	27,50%
2.000	286	0	286	14,30%
5.000	299	0	299	5,98%
10.000	309	0	309	3,09%
20.000	329	0	329	1,65%

FONTE: Platt (1998)

O número de vetores suportes tem um acréscimo considerável quando aplicado a $C = 0,1$ com kernel gaussiano de parâmetro $\delta^2 = 10$, como pode ser visto na tabela 6:

TABELA 6 – NÚMERO DE VS USANDO PRODUTO INTERNO NOS DADOS ARTIFICIAIS

Nº PONTOS TREINADOS	Nº VS $0 < \alpha < C$	Nº VS *BOUNDS*($\alpha = C$)	TOTAL VS	% VS
500	22	476	498	99,60%
1.000	82	888	970	97,00%
2.000	75	1.905	1.980	99,00%
5.000	30	4.942	4.972	99,44%
10.000	48	9.897	9.945	99,45%

FONTE: Platt (1998)

Analisando as tabelas, não é possível afirmar nada sobre o número de vetores suporte, em virtude de depender muito do banco de dados, da esparsidade dos dados e da importância da informação sobre a classificação.

Caso a maioria dos dados sejam vetores suporte, os dados estão classificados exageradamente, ocorrendo *overfitting*. Portanto, quanto menor o número de vetores suporte, melhor é a generalização na classificação. Como pode ser visto no exemplo utilizando o *kernel* gaussiano com $\sigma = 0,1$ o número de vetores suportes é de 29, tendo 30 pontos no treinamento, a separação está muito exagerada, como pode ser visto na figura 32:

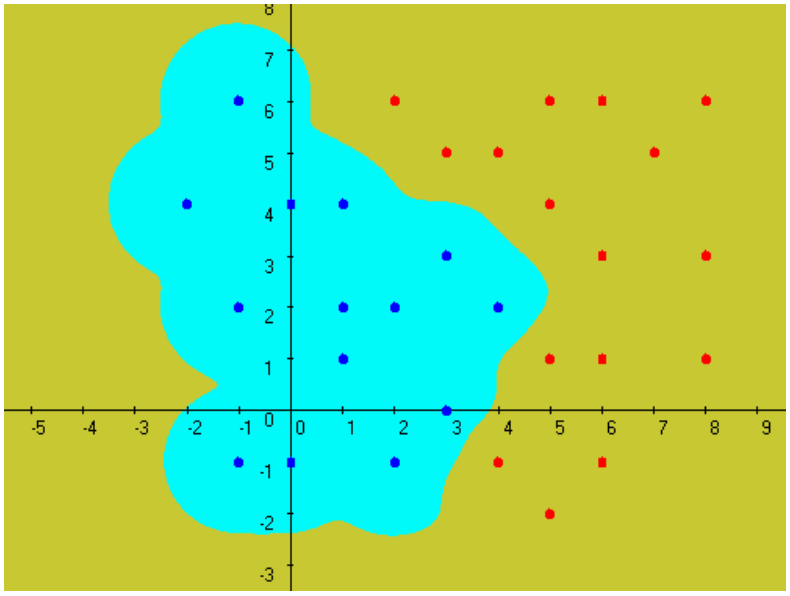

FIGURA 32: SEPARAÇÃO UTILIZANDO KERNEL GAUSSIANO
FONTE: A autora (2008)

Utilizando o kernel polinomial homogêneo de grau 3, tem-se uma generalização melhor, e o número de vetores suportes é reduzido para 5, como pode ser visto na figura 33:

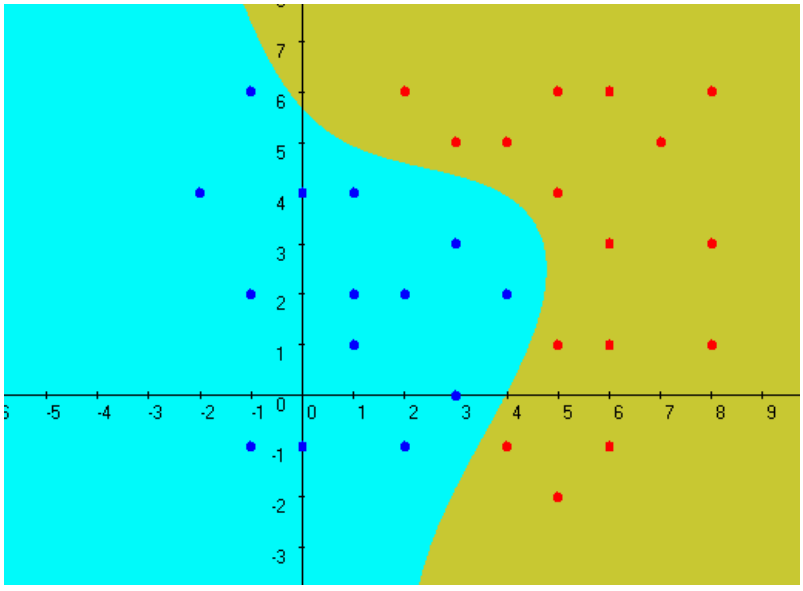

FIGURA 33: SEPARAÇÃO UTILIZANDO KERNEL HOMOGÊNEO
FONTE: A autora (2008)

O número de vetores suporte é uma fração do número de dados de treinamento. Treinando os dados com diferentes funções *kernel* e diferentes parâmetros, o conjunto

de vetores suporte solução de cada treinamento deve ter vetores suportes comuns com os outros conjuntos soluções.

Essa situação pode ser ilustrada com um exemplo no $\mathbb{R}^2$, tomando um conjunto de dados artificiais com 92 pontos e utilizando produto interno usual e $C = 1$, obtem-se uma classificação utilizando-se 22 vetores suporte e obtendo 11 pontos incorretos, devido à classificação ser linear e os pontos não serem linearmente separáveis, como pode ser visto na figura 33:

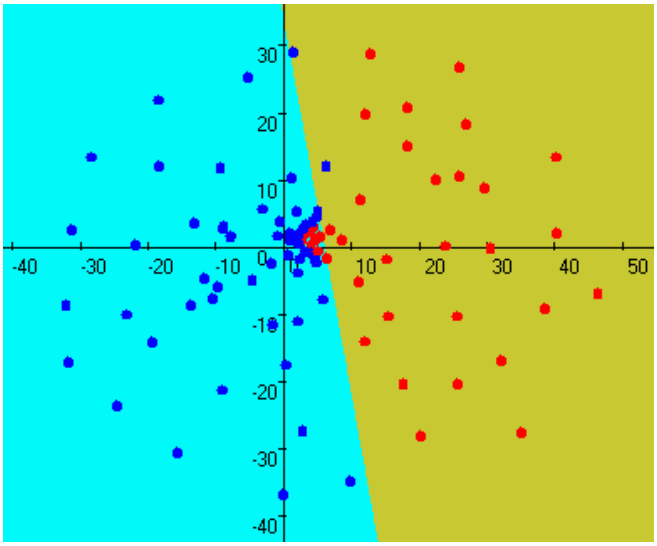

FIGURA 34 – SEPARAÇÃO UTILIZANDO PRODUTO INTERNO USUAL
FONTE: A autora (2008)

Utilizando *kernel* polinomial não homogêneo com parâmetros $p = 2$, $k = 1$ e $C = 1$, tem-se 20 vetores suportes e uma classificação com 13 pontos incorretos como pode ser visto na figura 34:

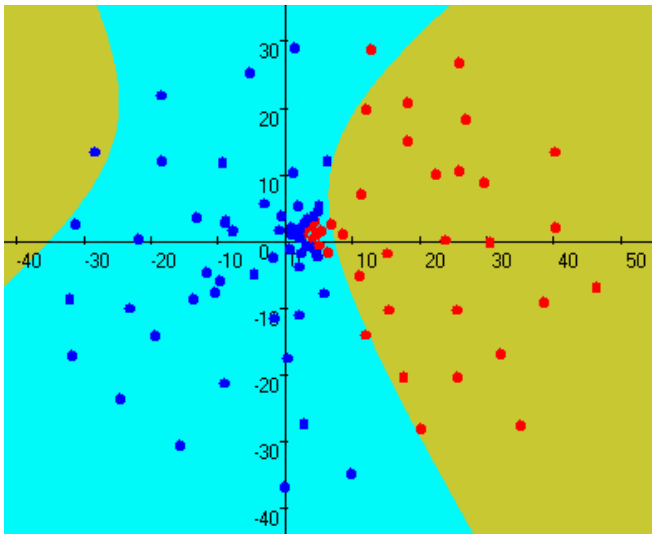

FIGURA 35 – SEPARAÇÃO UTILIZANDO KERNEL POLINOMIAL DE GRAU 2
FONTE: A autora (2008)

Utilizando *kernel* polinomial não homogêneo com parâmetros $p=3$, $k=1$ e $C=1$, obtem-se uma boa classificação com apenas 3 pontos incorretos, com 10 vetores suportes, como pode ser visto na figura 35:

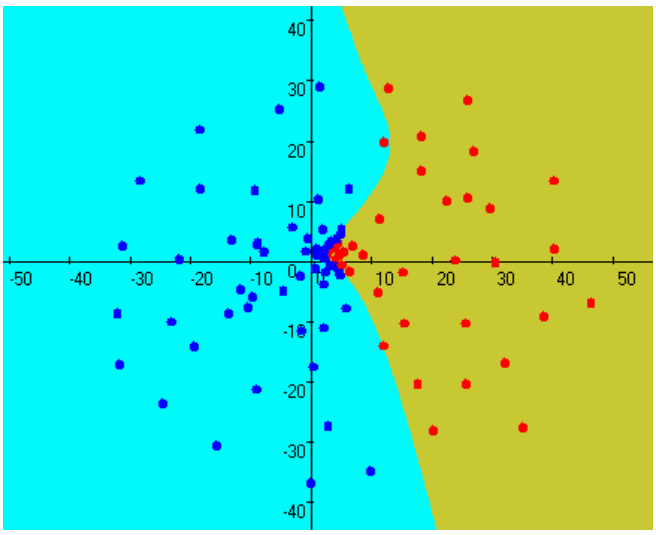

FIGURA 36 – SEPARAÇÃO UTILIZANDO KERNEL POLINOMIAL DE GRAU 3
FONTE: A autora (2008)

Nas três superfícies separadoras existem 5 vetores suportes comuns, os pontos x^3, x^5, x^{14}, x^{15} e x^{16}, esses pontos se encontram entre as duas classes como pode ser verificado na figura 36:

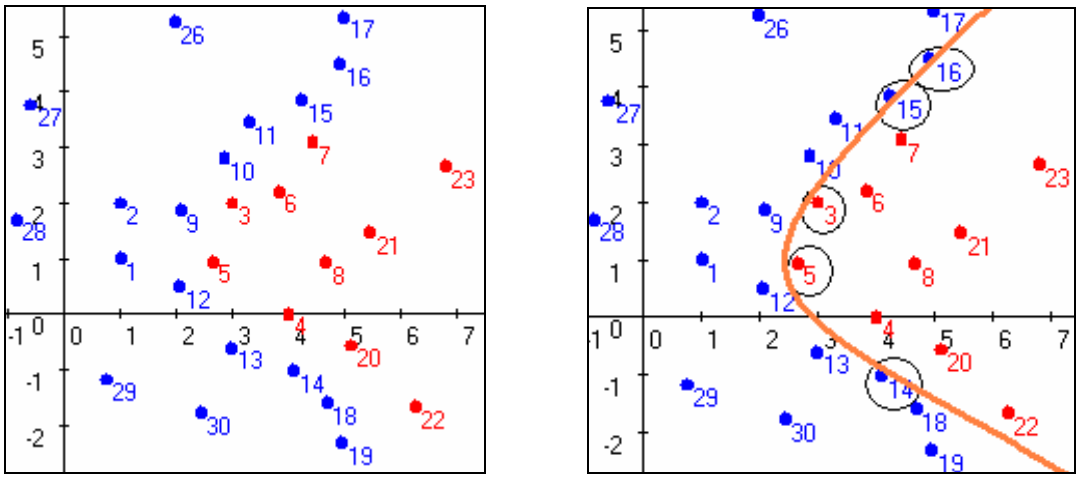

FIGURA 37 – VISUALIZAÇÃO NA REGIÃO DE SEPARAÇÃO
FONTE: A autora (2008)

Através das figuras 36, 37 e 38, ilustra-se a dificuldade de encontrar uma função *kernel* adequada para cada aplicação além da escolha dos parâmetros C e os específicos de cada *kernel*.

De conhecimento da fundamentação teórica torna-se possível a formulação do SVM e verifica-se a possibilidade de mais de um método de solução: o problema primal e o problema dual.

No próximo capítulo é apresentada a metodologia de resolução utilizada neste trabalho.

4 METODOLOGIA

Este capítulo abrange as soluções propostas para resolução do problema do *Support Vector Machine*. Especificamente é estudado o algoritmo do *Sequential Minimal Optimization* (SMO) proposto por John Platt.

Alguns dados utilizados foram extraídos da internet, alguns foram criados artificialmente e o banco de dados do problema de Icterícia foi fornecido pela professora Maria Teresinha Arns Steiner, da UFPR.

4.1 SOLUÇÕES DO MODELO DE PROGRAMAÇÃO QUADRÁTICA DO SVM

A modelagem do SVM envolve um problema não linear, quadrático e para resolver este problema existem alguns métodos clássicos de descida, como por exemplo, Método do Gradiente, que possui convergência linear[22] garantida para a função quadrática (FRIEDLANDER, 1994).

Outro método é o Método de Newton, que possui convergência quadrática[23], pois faz uso das derivadas segundas, tornando o processo caro, em termos de trabalho computacional. Se o número de variáveis for muito grande a memória necessária para armazenar as informações das derivadas pode ser insuficiente e este processo torna-se inviável (FRIEDLANDER, 1994).

O Método Quase-Newton possui ordem de convergência superlinear[24], sendo um método intermediário ao Método do Gradiente e ao Método de Newton. A vantagem em termos de trabalho computacional é que o número de operações é na ordem de n^2, no lugar de n^3 do Método de Newton (FRIEDLANDER, 1994).

---

[22] Convergência está relacionada com a velocidade de aproximar o erro cometido na aproximação do valor zero. A convergência Linear ocorre quando $e_{k+1} \leq re_k$, para $0 \leq r \leq 1$, onde e_k é o erro cometido na k-ésima iteração.

[23] Convergência Quadrática ocorre quando $e_{k+1} \leq r\left(e_k\right)^p$, para $r > 0$ e $p = 2$, e possui velocidade maior do que a convergência linear.

[24] Convergência Superlinear é intermediária a convergência linear e quadrática, ou seja, quando $k \to \infty$, $e_{k+1} / e_k = 0$.

Os Métodos de Filtro tem como principal idéia interferir o mínimo possível nas iterações de Programação Quadrática Seqüencial[25], mas de maneira que induza a convergência. A convergência é global, mas somente é garantida com a escolha de um passo eficiente. Esses métodos possuem um sério problema, a manipulação do parâmetro de penalidade, pois dependendo do valor pode levar a solução ótima ou ficar bastante lento (RIBEIRO, 2005).

Os métodos descritos acima podem obter as soluções ótimas após certo número de iterações. Existem pacotes computacionais que facilitam a utilização do SVM, como o MINOS, do *Stanford Optimization Laboratory*, com uma estratégia híbrida, o LOQO, que utiliza o método do ponto interior, e o MATLAB *Optimization Toolbox*, que utiliza uma sub-rotina de programação quadrática. Também existem os pacotes como o SVM^{light} de Joachims, um pacote da Royal Holloway, University of London.

O Lingo, ferramenta computacional, pode ser utilizado para resolução de problemas não lineares como o problema de modelagem do SVM. O Lingo encontra a solução exata do problema.

Chih-Chung Chang e Chih-Jen Lin propuseram o LIBSVM, disponível no site http://www.csie.ntu.edu.tw/~cjlin/libsvm/ , que é um *sofware* integrado do *Support Vector Machine* para classificação, regressão e multi-classificação e também possui uma biblioteca de dados.

Optou-se por uma heurística para a solução do SVM usando o *Sequential Minimal Optimisation* (SMO), proposto por John Platt (PLATT, 1998), em virtude de contemplar o modelo original do SVM.

4.1.1 Lingo

O Lingo é uma ferramenta computacional para modelagem de problemas lineares e não lineares de otimização. O processo de otimização consiste em tentar encontrar a melhor solução possível para um dado problema, usando técnicas de programação matemática.

Os problemas lineares podem ser resolvidos de forma mais rápida e com maior precisão do que modelos não lineares.

[25] Programação Quadrática Seqüencial é um método para resolução de problemas não lineares, através da resolução de uma seqüência de Problemas Quadráticos, visto que, próximo de um ponto de mínimo (ou máximo) a aproximação é quadrática.

Para problemas representados por modelos lineares, quando resolvidos completamente pelo Lingo, tem-se a garantia de que o valor encontrado é a melhor solução, ou seja, a solução ótima. O mesmo não ocorre no caso de modelos não lineares, pois a solução pode permanecer presa em um ótimo local, não obtendo a melhor solução.

O SVM é um modelo não linear, quadrático e convexo, com garantia de que o ótimo local também é o ótimo global. Portanto, pode ser resolvido pelo Lingo. No entanto, o Lingo possui restrição no número de variáveis, tornando-se inviável para banco de dados com milhares de pontos.

4.1.2 *Sequential Minimal Optimization*

O *Sequential Minimal Optimization* (SMO) é um algoritmo heurístico que utiliza apenas duas variáveis em cada iteração. Com isso apresenta uma solução analítica na iteração, além de não haver a necessidade de resolver o problema quadrático.

Trabalhando com dois Multiplicadores de Lagrange de cada vez e mantendo os outros fixos, a condição principal para o SMO é $\sum_{i=1}^{l} \alpha_i y_i = 0$, ou equivalentemente:

$$\sum_{i \in A} \alpha_i y_i = \sum_{j \in B} \alpha_j y_j \tag{3.34}$$

Ou seja, essa condição obriga que quando um Multiplicador é atualizado o outro deve ser ajustado para manter a condição verdadeira.

A escolha dos dois pontos é feita a partir de uma heurística, já a atualização dos valores é feita de forma analítica (CRISTIANINI e SHAWE-TAYLOR, 2000).

Para não violar a condição de $\sum_{i=1}^{l} \alpha_i y_i = 0$, os novos Multiplicadores de Lagrange devem respeitar:

$$\alpha_1^{new} y_1 + \alpha_2^{new} y_2 = \text{ constante } = \alpha_1^{old} y_1 + \alpha_2^{old} y_2 \tag{3.35}$$

O algoritmo primeiramente encontra o valor para α_2^{new}, e utiliza-o para obter o valor de α_1^{new}.

Para que os novos valores sejam viáveis para a resolução do problema deve-se respeitar as restrições: $0 \leq \alpha_1$, $\alpha_2 \leq C$, como ilustra a figura 37:

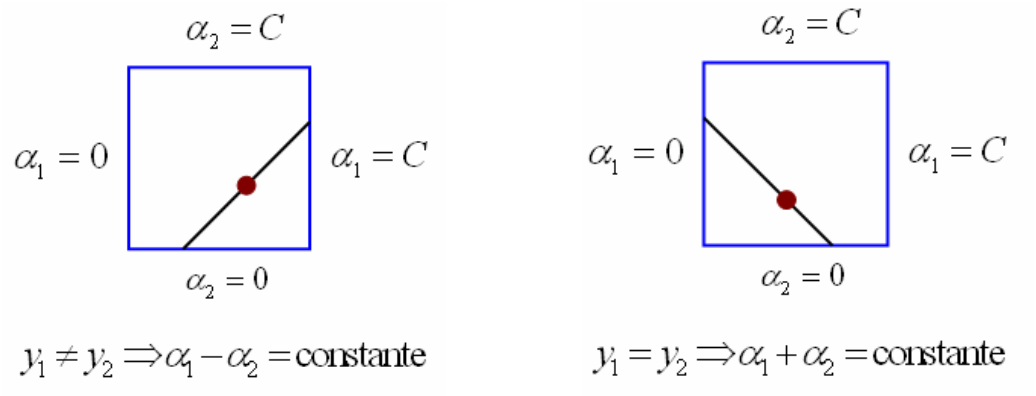

FIGURA 38 – LIMITAÇÕES DOS MULTIPLICADORES DE LAGRANGE
FONTE: Platt (1998)

Podendo ser restrito pela condição:

$$L \leq \alpha_2^{new} \leq H \tag{3.36}$$

onde:

se $y_1 \neq y_2$

$$L = \max\left\{0, \alpha_2^{old} - \alpha_1^{old}\right\}$$

$$H = \min\left\{C, C - \alpha_1^{old} + \alpha_2^{old}\right\}$$

se $y_1 = y_2$

$$L = \max\left\{0, \alpha_1^{old} + \alpha_2^{old} - C\right\}$$

$$H = \min\left\{C, \alpha_1^{old} + \alpha_2^{old}\right\}$$

Onde α_i^{new} é o novo valor do Multiplicador de Lagrange do ponto x^i e α_i^{old} é o valor anterior. O valor da função em x^i que denota a função atual determinada pelos valores dos Multiplicadores de Lagrange e por b no estágio atual da aprendizagem é dado por:

$$f\left(x^i\right) = \sum_{j=1}^{l} \alpha_j y_j K\left(x^j, x^i\right) + b \tag{3.37}$$

O valor E_i determina a diferença de $f\left(x^i\right)$ e o padrão y_i a que pertence o ponto x^i, ou seja, é a distancia do ponto ao hiperplano atual dado pela atualização dos Multiplicadores de Lagrange, é dado por:

$$E_i = f\left(x^i\right) - y_i = \left[\sum_{j=1}^{l} \alpha_j y_j K\left(x^j, x^i\right) + b\right] - y_i \tag{3.38}$$

Este valor pode ser grande quando o ponto está classificado corretamente, por exemplo, se o padrão $y_1 = 1$ e $f\left(x_1\right) = 5$, então $E_1 = 4$.

A quantidade adicional exigida é a segunda derivada da função objetivo ao longo da linha diagonal, que pode ser expressa por κ, definido por:

$$\kappa = K\left(x^1, x^1\right) + K\left(x^2, x^2\right) - 2K\left(x^1, x^2\right) = \left\|\phi\left(x^1\right) - \phi\left(x^2\right)\right\|^2 \tag{3.39}$$

onde:

x^1 e x^2 são os pontos associados a α^1 e α^2, respectivamente.

O máximo valor da função objetivo será obtido com o valor:

$$\alpha_2^{new} = \begin{cases} H, \text{ se } \alpha_2^{aux} > H \\ \alpha_2^{aux}, \text{ se } L \le \alpha_2^{aux} \le H \\ L, \text{ se } \alpha_2^{aux} < L \end{cases} \tag{3.40}$$

Sendo α_2^{aux} um valor truncado, ou seja, limitado por $L \le \alpha_2^{aux} \le H$, dado por:

$$\alpha_2^{aux} = \alpha_2^{old} + \frac{y_2\left(E_1 - E_2\right)}{\kappa} \tag{3.41}$$

Através do valor de α_2^{new}, encontra-se α_1^{new} dado por:

$$\alpha_1^{new} = \alpha_1^{old} + y_1 y_2\left(\alpha_2^{old} - \alpha_2^{new}\right) \tag{3.42}$$

O SMO usa dois critérios para selecionar dois pontos ativos, ou seja, $0 < \alpha < C$, para garantir que a função objetivo aproveite um grande acréscimo, na atualização dos valores.

Têm-se duas heurísticas, uma para a escolha de α_1 e outra para escolha de α_2. Na primeira heurística, o ponto x^2 é escolhido entre os pontos que violam as condições de KKT. O algoritmo atravessa por todo o conjunto de dados de treinamento que violam as condições de KKT e seleciona um para atualizar, isso é feito através de um dos testes $E_2 \cdot y_2 < -tol$ e $\alpha_2 < C$ ou $E_2 \cdot y_2 > tol$ e $\alpha_2 > 0$. Quando tal ponto é encontrado, utiliza-se a segunda heurística para selecionar o ponto x^1, que deve ser escolhido de tal maneira que seja atualizado com x^2, causando uma grande mudança, que deve resultar em um grande acréscimo na função objetivo dual.

Para encontrar um bom ponto sem muitos cálculos, uma heurística rápida escolhe x^1, maximizando o valor dado por $|E_1 - E_2|$, se E_2 é positivo, o SMO escolhe o x^1 com o menor E_1, e se E_2 é negativo, então o SMO escolhe x^1 com o maior E_1.

Se esta escolha falhar em obter um acréscimo significante na função objetivo dual, o SMO experimenta cada ponto x^1 que tenha valores de α diferente dos limites, ou seja, $0 < \alpha < C$, começando aleatoriamente. Se ainda não houver progresso significativo, o SMO procura por todo o conjunto de dados de treinamento para encontrar um ponto x^1 adequado.

Se ocorrer alteração de valores, a heurística retorna para escolher outros pontos x^2 e x^1, até que todos os pontos estejam obedecendo as condições de KKT. Caso contrário termina o processo.

A lista dos erros de todos os pontos de treinamento é mantida na memória para reduzir contas adicionais.

A solução satisfaz as condições de complementaridade de Karush-Kuhn-Tucker que para classificar com a máxima margem entre os conjuntos deve obedecer $\alpha_i \left[y_i \left(\left\langle w \cdot x^i \right\rangle + b \right) - 1 \right] = 0$ para $i = 1, 2, ..., l$, ou seja, $\alpha_i \left[y_i f\left(x^i\right) - 1 \right] = 0$. Correspondente a:

$$\begin{aligned} \alpha_i = 0 \quad &\Leftrightarrow y_i f\left(x^i\right) \geq 1 \\ 0 < \alpha_i < C \quad &\Leftrightarrow y_i f\left(x^i\right) = 1 \\ \alpha_i = C \quad &\Leftrightarrow y_i f\left(x^i\right) \leq 1 \end{aligned} \tag{3.43}$$

Para melhorar a velocidade de convergência, é possível uma escolha do conjunto dos dois pontos a serem otimizados, baseados na suas respectivas contribuições para progresso geral em direção à solução. Se a quantidade de contas exigidas para implementar a estratégia de selecionar o conjunto reduzir o número de iterações, então obtem-se uma taxa de convergência melhor.

Cristianini e Shawe-Taylor (2000) descreve três critérios de parada, que são detalhados a seguir:

1. Monitoramento da função objetivo dual. A função objetivo dual quadrático para um problema de otimização de SVM consegue o máximo na solução. Monitorando o valor da função, especificamente o valor do crescimento a cada passo, surge um critério de parada. O treinamento para quando a taxa de crescimento da função objetivo for menor que certa tolerância, por exemplo, 10^{-9}. Infelizmente, este critério não é confiável e em alguns casos, mostrou resultados pobres.

2. Monitoramento das condições de KKT para o problema primal. A condição de Karush-Kuhn-Tucker (KKT) é necessária e suficiente para encontrar o ponto ótimo no problema quadrático, sendo definida como um critério natural, dado pelas condições 3.43, onde $0 \leq \alpha_i \leq C$. Naturalmente, este critério deve novamente ser verificado com certa tolerância, por exemplo, 10^{-2}.

3. Outra maneira para caracterizar a solução é por meio do *gap* entre as funções objetivas: dual e primal. Porém, esse critério de parada é válido apenas quando se tem um hiperplano linear.

Resolvendo o SMO, o valor de b, do vetor w e dos erros E_i são calculados separadamente. Após cada iteração, em que as condições de KKT são satisfeitas para ambos os pontos x^1 e x^2. Os valores podem ser atualizados analisando sempre o valor atual com o anterior da função $f\left(x^i\right) = \sum_{j=1}^{l} \alpha_j y_j K\left(x^j, x^i\right) + b$.

O valor de b para o ponto x^1 é definido por b_1, o qual deve forçar a saída do SVM para y_1 quando a entrada for o ponto x^1, dado por:

$$b_1 = -\left[E_1 + y_1 \left(\alpha_1^{new} - \alpha_1^{old} \right) K\left(x^1, x^1 \right) + y_2 \left(\alpha_2^{new} - \alpha_2^{old} \right) K\left(x^1, x^2 \right) + b^{old} \right] \quad (3.44)$$

O valor de b para o ponto x^2 é definido por b_2, o qual deve forçar a saída do SVM para y_2 quando a entrada for o ponto x^2, dado por:

$$b_2 = -\left[E_2 + y_1 \left(\alpha_1^{new} - \alpha_1^{old} \right) K\left(x^1, x^2 \right) + y_2 \left(\alpha_2^{new} - \alpha_2^{old} \right) K\left(x^2, x^2 \right) + b^{old} \right] \quad (3.45)$$

Se os valores b_1 e b_2 são iguais, então este será o novo valor de b, ou seja, $b^{new} = b_1 = b_2$. Caso contrário, o intervalo entre b_1 e b_2 são todos os *thresholds* que são consistentes com as condições de KKT, portanto, o SMO escolhe o *threshold* que está no meio do intervalo, ou seja:

$$b^{new} = \frac{b_1 + b_2}{2} \quad (3.46)$$

Quando os dados são separados linearmente, pode-se atualizar o valor do vetor w por:

$$w^{new} = w^{old} + y_1 \left(\alpha_1^{new} - \alpha_1^{old} \right) \vec{x}^1 + y_2 \left(\alpha_2^{new} - \alpha_2^{old} \right) \vec{x}^2 \quad (3.47)$$

Os erros E_i são atualizados a cada iteração por:

$$E_i^{new} = E_i^{old} + y_1 \left(\alpha_1^{new} - \alpha_1^{old} \right) K\left(x^1, x^i \right) + y_2 \left(\alpha_2^{new} - \alpha_2^{old} \right) K\left(x^2, x^i \right) + b^{new} - b^{old} \quad (3.48)$$

O objetivo da heurística SMO é obter os valores dos Multiplicadores de Lagrange para que estes tenham os erros tendendo a zero, é incorreto afirmar que $E_1 = 0$ e $E_2 = 0$, pois nas primeiras iterações, os valores dos multiplicadores não estão ótimos, eles ainda poderão ser atualizados, portanto pode ter erro ainda.

Atualização da função objetivo pode ser feita pelo *gap*, que é a diferença entre a função objetivo atual e a anterior:

$$gap = \left(\alpha_1^{new} - \alpha_1^{old}\right) + \left(\alpha_2^{new} - \alpha_2^{old}\right) - \left[\sum_{j=1}^{2}\sum_{i=1}^{l} y_j y_i \left(\alpha_j^{new} - \alpha_j^{old}\right) K\left(x^j, x^i\right)\right]_{\substack{i\neq 1 \\ i\neq 2}} +$$
$$-\frac{1}{2} y_1^2 \left(\alpha_1^{new} - \alpha_1^{old}\right) K\left(x^1, x^1\right) - \frac{1}{2} y_2^2 \left(\alpha_2^{new} - \alpha_2^{old}\right) K\left(x^2, x^2\right) + \tag{3.49}$$
$$- y_1 y_2 \left[\alpha_1^{new}\alpha_2^{new} - \alpha_1^{old}\alpha_2^{old}\right] K\left(x^1, x^2\right)$$

O SMO possui uma rotina de laços (*loops*), forçando a heurística a procurar por todos os pontos de treinamento os quais estão infringindo as condições de KKT.

A seguiir o fluxograma da heurística (Figura 39):

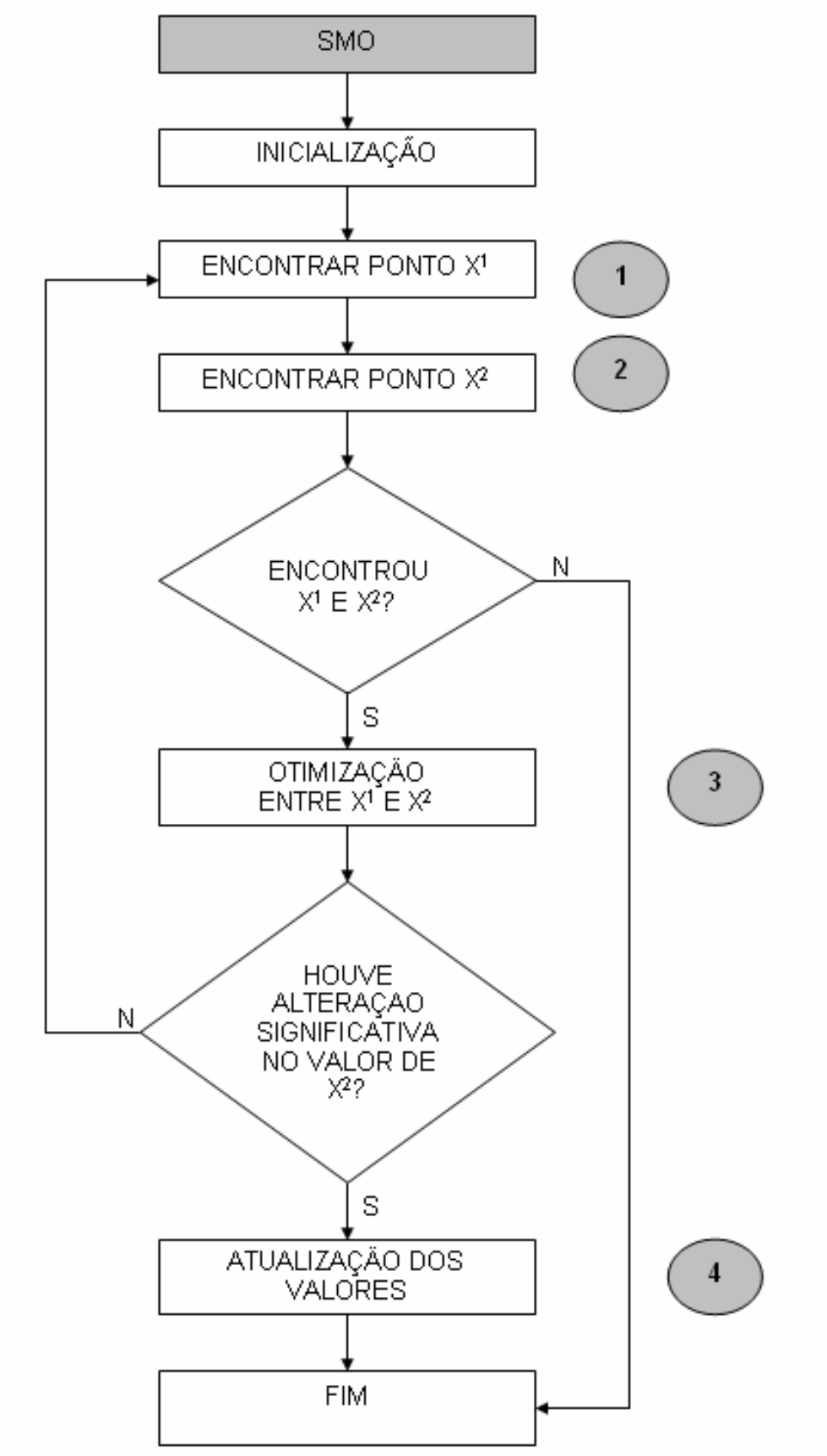

FIGURA 39 – FLUXOGRAMA DA ESCOLHA DO PRIMEIRO PONTO

FONTE: A autora (2008)

1 - Primeira heurística:

O ponto x^2 é escolhido entre os pontos que violam as condições de KKT, através de um dos testes $E_2 \cdot y_2 < -tol$ e $\alpha_2 < C$ ou $E_2 \cdot y_2 > tol$ e $\alpha_2 > 0$. Primeiramente entre os pontos x^i cujos $\alpha_i \neq 0$ e $\alpha_i \neq C$, posteriormente, entre os quais $\alpha_i = 0$.

2 - Segunda heurística:

O ponto x^1 é escolhido como o ponto que tem a maior diferença entre os erros, ou seja, o maior valor $|E_1 - E_2|$. Se esta escolha falhar em obter um acréscimo significante, o SMO escolhe pontos x^1 cujo valor do multiplicador de lagrange esteja entre 0 e C, ou seja, $0 < \alpha < C$, aleatoriamente. Caso não ocorra progresso significativo, o SMO procura por todo o conjunto de treinamento para encontrar x^1 adequado, a escolha é aleatória.

3 - Otimização:

Primeiramente é verificado se os pontos x^1 e x^2 não são iguais, caso sejam, o algoritmo volta a escolher novos pontos x^1 e x^2. Calcula-se os valores limitantes inferior L e superior H, caso sejam iguais o algoritmo volta a escolher pontos x^1 e x^2. Caso contrário, calcula-se o valor de κ, e atualiza-se o valor do multiplicador de lagrange α_2^{new} (equação 3.41), verificando se o valor está entre os limitantes.

4 - Atualização:

Se ocorrer atualização significativa em relação ao novo valor de α_2, então ocorre a atualização dos valores de α_1, α_2, E (equação 3.48), b (equação 3.46) e da função objetivo (equação 3.49).

A finalização do algoritmo ocorre quando todos os pontos não tiverem mais alteração de valores, ou seja, quando nenhum x^1 for escolhido.

4.1.3 Visual Basic

O *Microsoft Visual Basic* é uma linguagem para desenvolvimento de aplicações visuais para ambiente *Windows* baseado em *Basic* (*Beginners All-purpose Symbolic*

Instruction Code). Possui uma interface gráfica, verificando automaticamente a sintaxe de comandos, recursos avançados de compilação e rastreamento de erros.

O *Visual Basic* tem um ambiente de desenvolvimento que aborda cada aspecto da programação, de aplicativos educacionais à programação de banco de dados, e de aplicativos financeiros ao desenvolvimento de componentes da Internet (PETROUTSOS, 1999).

O *Visual Basic* foi utilizado para programar o processo de reconhecimento de padrões utilizando o algoritmo SMO.

4.2 PROGRAMAÇÃO DO ALGORITMO SMO

Para treinar e testar os dados, o algoritmo SMO foi programado em linguagem *Visual Basic*, possibilitando a verificação de todo o processo e a visualização das superfícies de separação, quando os dados estavam no plano cartesiano.

A interface do programa possibilita a visualização dos pontos, quando os mesmos estão no $\mathbb{R}^2$, como mostra a figura 40:

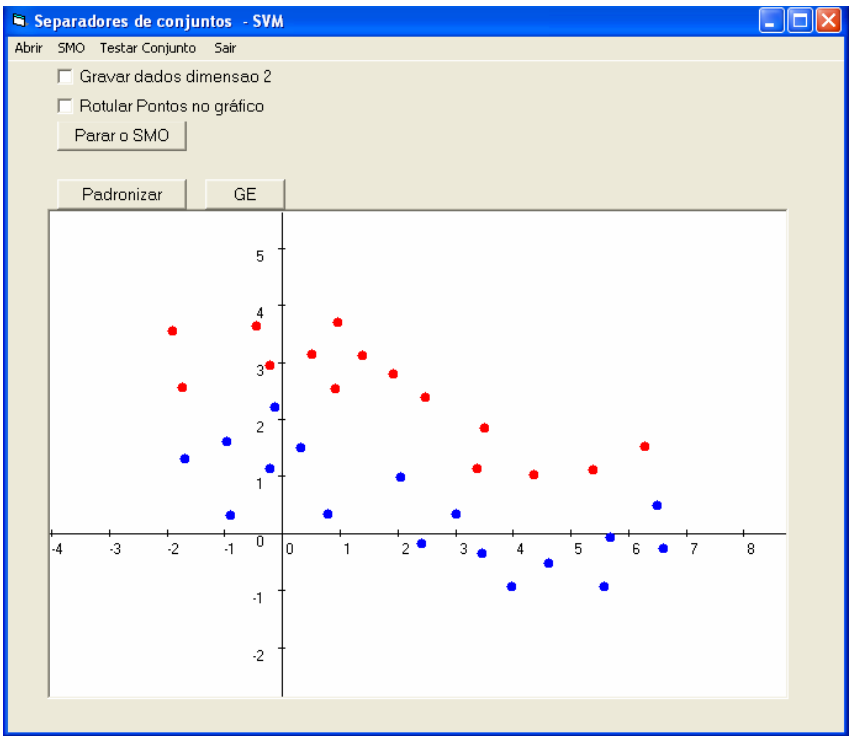

FIGURA 40 – PÁGINA INICIAL DO PROGRAMA

Fonte: A autora (2008)

Quando o programa carrega o banco de dados pode ser iniciada a escolha dos parâmetros da modelagem dual do SVM, através do lagrangeano. O Usuário escolhe a função *kernel*, os parâmetros do *kernel* escolhido, o parâmetro C e aperta o botão **Confirmar a escolha e continuar** para que o programa inicialize o algoritmo SMO, como mostra a figura 41:

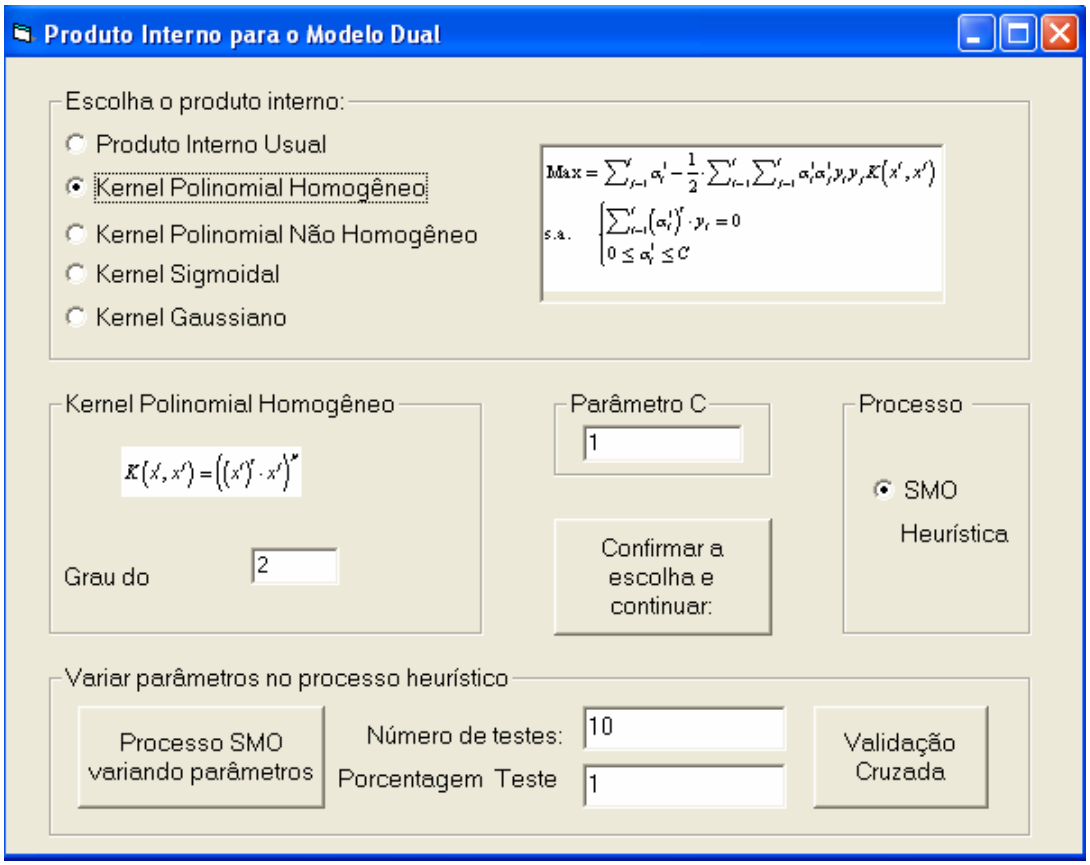

FIGURA 41 – TELA DE ESCOLHA DA FUNÇÃO KERNEL E DOS PARÂMETROS
Fonte: A autora (2008)

Caso o usuário queira avaliar todos os parâmetros e todas as funções *kernel*, não necessita escolher nenhum parâmetro, apenas inicializar o botão **Processo SMO variando parâmetros**. Automaticamente, o programa irá executar o algoritmo SMO.

Se o objetivo for avaliar os parâmetros escolhidos, o usuário deve escolher o *kernel*, os parâmetros, digitar o número de testes que deseja realizar e a porcentagem do conjunto de dados que deve ser reservado para testes. Inicializando o processo no botão **Validação Cruzada**.

Quando os dados estão no $\mathbb{R}^2$, o programa executa o algoritmo e plota a superfície separadora. Como o processo utilizando o *kernel* faz um mapeamento

implícito em alguma dimensão maior, para fazer a classificação dos dados em regiões distintas, o programa calcula a função de decisão descrita por:

$$f\left(x^i\right)=\sum_{j=1}^{l}\alpha_j y_j K\left(x^j,x^i\right)+b \qquad (3.50)$$

Ou seja, para cada ponto do plano cartesiano, a função decisão dependerá do valor dos vetores suporte e do próprio ponto. E depois de realizado o treinamento a resposta pode ser visualizada através das regiões discriminadas nas cores azul e amarela, como mostra a figura 42:

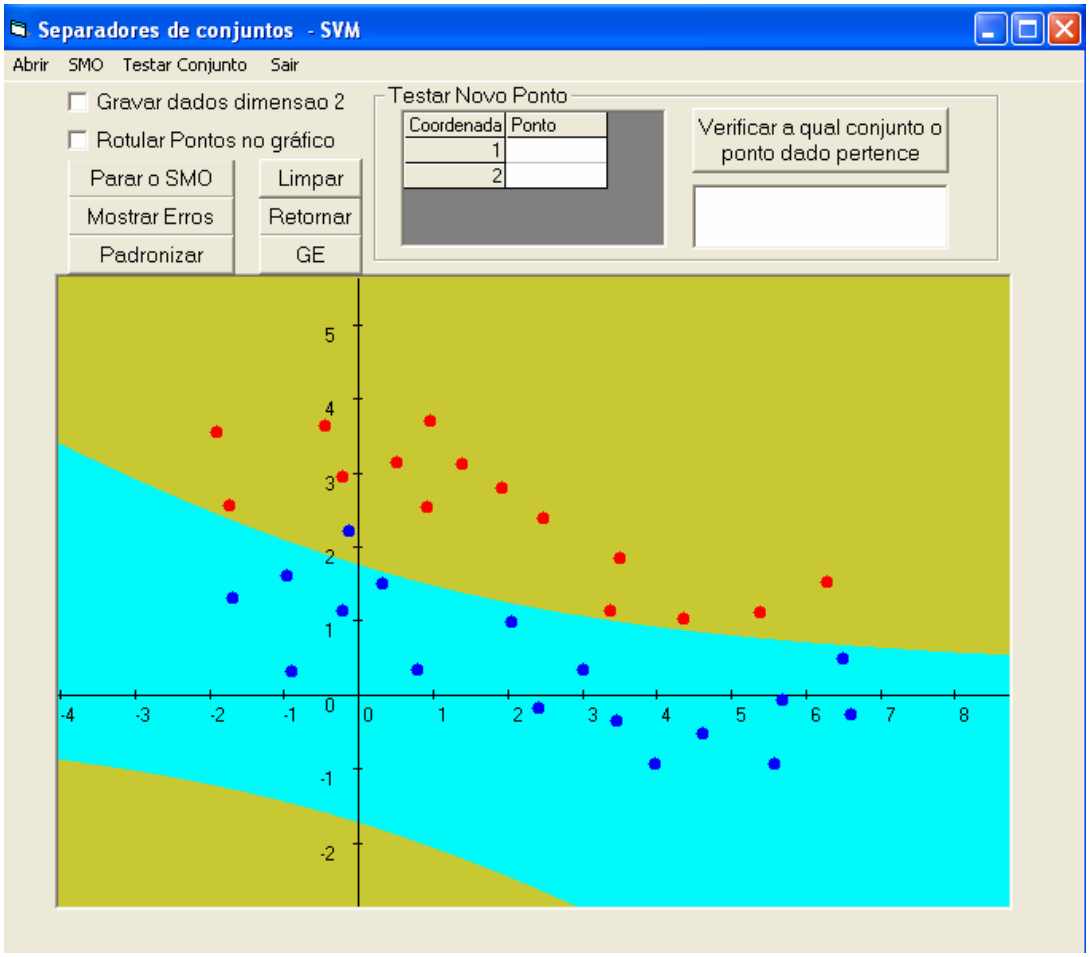

FIGURA 42 – VISUALIZAÇÃO DA RESPOSTA COM DADOS DE DIMENSÃO 2
FONTE: A autora (2008)

Os dados de treinamento como informações dos pontos, dos multiplicadores de lagrange, os parâmetros e os erros são gravados em um arquivo (de extensão “.txt”) para posterior consulta.

O usuário pode verificar os erros na classificação utilizando o botão **Mostrar Erros** como mostra a figura 43:

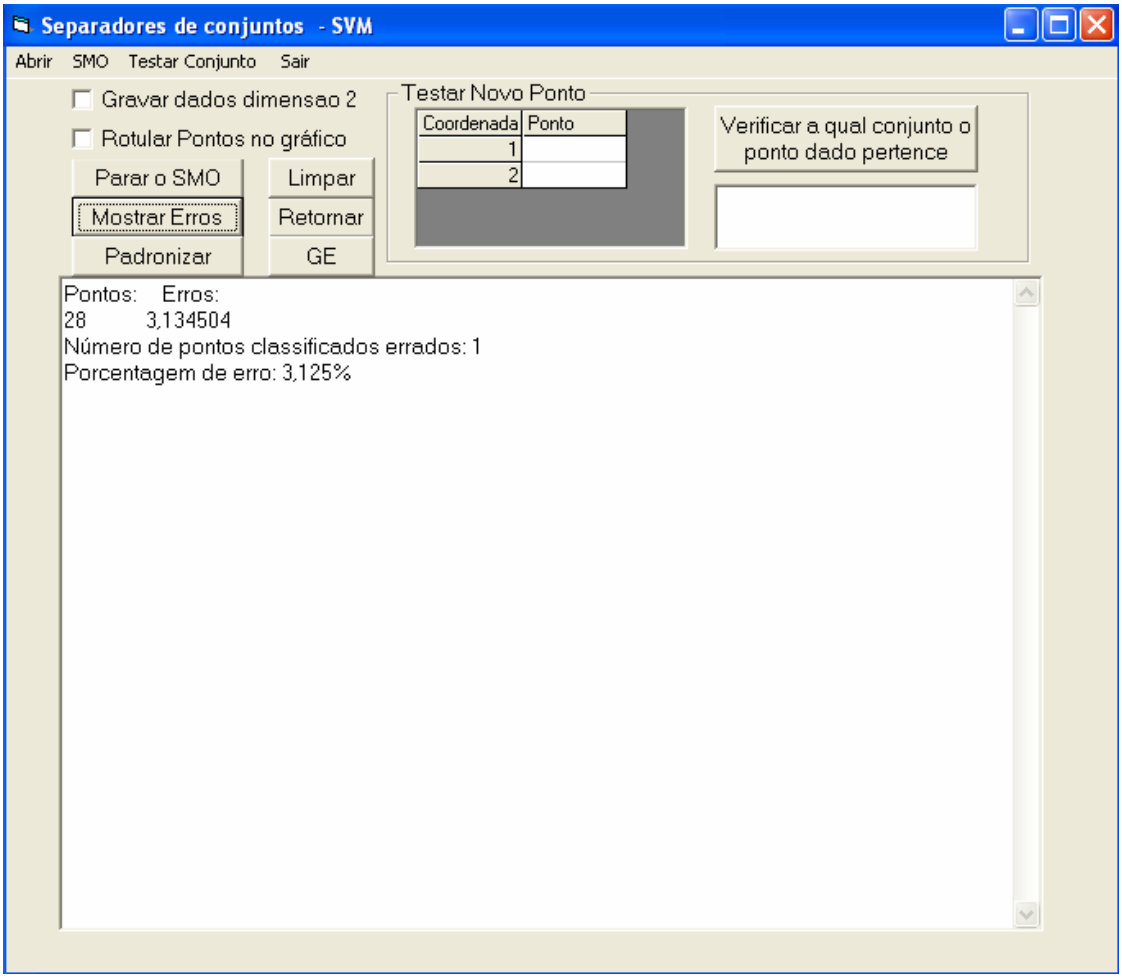

FIGURA 43 – VISUALIZAÇÃO DOS ERROS
FONTE: A autora (2008)

Caso os dados sejam de dimensão maior que dois, não é possível a visualização. O programa irá mostrar quais são os pontos classificados incorretamente em relação aos parâmetros escolhidos, conforme mostra a figura 43.

Terminado o processo de classificação é possível verificar como um ponto ou um determinado conjunto de dados é classificado. Um ponto é classificado colocando as coordenadas em **Testar novo ponto**. Caso os dados tenham dimensão dois, é possível visualizar o ponto no plano cartesiano. Caso contrário, o programa apenas irá informar a qual conjunto o dado pertence. Se o objetivo é testar um conjunto de pontos, isso pode ser realizado através do menu **testar conjunto**, o qual abrirá o banco de dados de teste e informará no final dos testes, a porcentagem de pontos classificados incorretamente.

As informações do teste são gravadas em um arquivo (de extensão “.txt”) para consultas posteriores.

Esse programa possibilita a análise das respostas e a comparação entre as diversas funções kernel e seus parâmetros.

4.3 COMPARAÇÃO ENTRE O ALGORITMO SMO E O LINGO

Testando um mesmo conjunto de dados no SMO e no Lingo temos a seguinte separação em ambos os processos, verificada na figura 44:

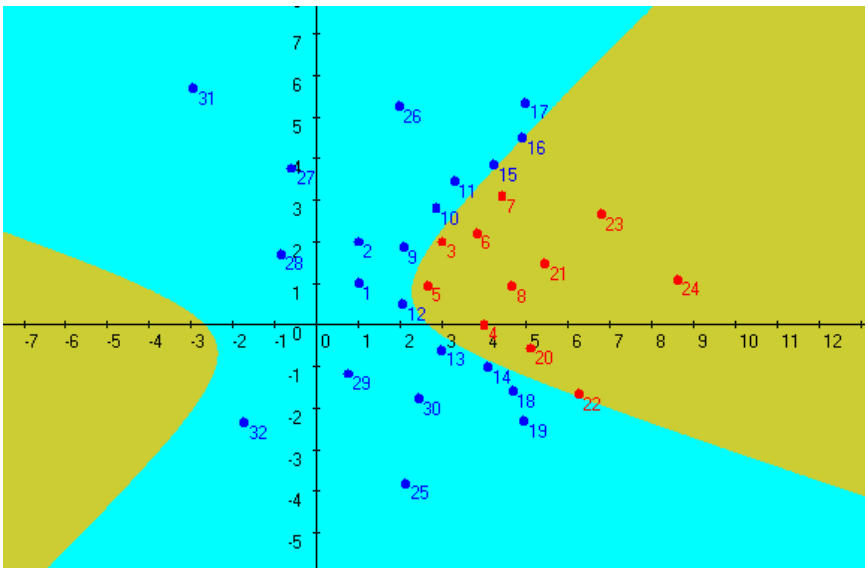

FIGURA 44 – SEPARAÇÃO ÓTIMA
FONTE: A autora (2008)

A função *kernel* utilizada foi o polinomial homogêneo de grau 2, o número de vetores suportes são 6, sendo que 2 são vetores suportes *bounds*.

O tempo computacional utilizando o Lingo para resolução deste pequeno exemplo é em média 0,2589 segundos, já o tempo computacional utilizando o SMO é em média 0,4549 segundos. O Lingo tem vantagem no tempo computacional para esse problema.

No entanto, quando os problemas são maiores, o Lingo apresenta problemas de instabilidade numérica, pois a matriz de *kernel* se torna densa, e as soluções

apresentadas são locais. Não sendo uma ferramenta adequada para a resolução de problemas com muitos pontos.

4.4 DADOS

Um banco de dados com aplicações reais pode ser localizado no site: ftp://ftp.ics.uci.edu/pub/machine-learning-databases/. Esses dados precisam de tratamento, ou seja, para serem adequados a um processo de classificação.

Chih-Chung Chang e Chih-Jen Lin disponibilizaram uma biblioteca de dados, LIBSVM, e também um software para classificação por *Support Vector Machine* disponível em http://www.csie.ntu.edu.tw/~cjlin/libsvm. Esses dados já estão configurados para aplicação direta da ferramenta de reconhecimento de padrões.

Os dados escolhidos para teste e comparação de resultados foram: Icterícia, *Adult*, *German* e *Australian*. Sendo que sobre *German* e *Australian* não constam informações no site.

Para os bancos de dados que não possuíam banco de testes foi realizada a validação cruzada *holdout*, na qual 20% dos dados de treinamento eram separados para realização de testes. Esse percentual de dados foi escolhido aleatoriamente, tomando o cuidado de manter a proporção de dados das duas classes.

4.4.1 Icterícia

Os dados do diagnóstico médico de Cálculo e Câncer foram disponibilizados pela Professora Maria Teresinha Ans Steiner, encontram-se no apêndice 1.

O problema médico, referenciado por Steiner (1995), trabalha com a icterícia, que representa o sintoma da cor amarelada da pele e das mucosas, que pode ser causada por várias doenças. Essas doenças podem ser separadas em dois grupos: Colestase[26] e outras causas[27]. O grupo Colestase é subdividido em dois grupos: os que possuem obstrução no fluxo biliar por cálculo e por câncer.

[26] Colestase: é quando existe dificuldades ou impedimento do fluxo biliar, do fígado para o intestino.
[27] Outras causas são distúrbios como anemias e hepatites.

Os dados foram coletados em 118 pacientes, sendo 35 com câncer e 83 com cálculo. As características são 14, sendo elas: idade, sexo e dados de exames médicos como: Bilirrubina[28], Albumina[29], entre outros.

As técnicas utilizadas para classificação desses dados foram: Programação Linear, Discriminante de Fischer, Método K-Vizinhos, Regressão Logística e Redes Neurais. Os resultados obtidos com os dados normalizados usando as 14 características foram:

TABELA 7 – COMPARAÇÃO DE RESULTADOS NOS DADOS ICTERÍCIA

TÉCNICA DE RECONHECIMENTO DE PADRÕES	ERROS DE TREINAMENTO	ERROS DE TESTE
Programação Linear	16,35%	24,99%
Discriminante de Fisher	18,87%	19,44%
Método K-Vizinhos	22,95%	25,00%
Regressão Logística	16,67%	19,44%
Redes Neurais	23,00%	27,97%

FONTE: Steiner (1995)

Os dados foram analisados por um especialista na área, e descartou-se 7 dados, considerados atípicos. Com o banco de dados contendo 111 pontos os resultados foram melhores do que os apresentados na tabela anterior.

4.4.2 *Adult*

Os dados *Adult* foram extraídos do banco de dados do censo encontrado em Http://www.census.gov/ftp/pub/DES/www/welcome.html, doado por Ronny Kohavi e Barry Becker em 1994. Inicialmente eram 48.842 casos, de dados contínuos e discretos, dos quais foram retirados valores desconhecidos, duplicados e conflitantes e divididos em dois grupos, um para treinamento e outro para teste.

As características são: idade, classe trabalhadora, peso, educação, estado civil, profissão, relacionamento[30], raça, sexo, ganho de capital, perda de capital, horas de

[28] Bilirrubina mede a intensidade da cor amarelada, o aumento do nível de bilirrubina reflete na cor amarelada.
[29] Albumina é uma proteína sintetizada exclusivamente no fígado, está presente no sangue, no leite e na clara de ovo.
[30] Relacionamento com os familiares, ou seja, com quem vive.

trabalho semanais, nacionalidade. A saída é determinar se a pessoa tem rendimento inferior a U$ 50.000 ou superior a essa quantia, saída binária. Essas 14 características foram armazenadas em vetores com 123 coordenadas, devido às variáveis discretas.

Os dados *Adult* foram utilizados para treinamento e teste por vários métodos, sendo a precisão relatada de aproximadamente 85% e uma média de 16,24% de erro nos testes.

4.5 PARÂMETROS

Para treinar o modelo com os bancos de dados escolhidos foram escolhidos alguns parâmetros. A escolha do parâmetro C, valor de importância ao erro na classificação, é determinado por $C = 10^{-t}$, sendo $t = -4, -3, ..., 3$.

As funções *kernel* escolhidas para o treinamento dos pontos e seus respectivos valores são:

- *Kernel* Polinomial não homogêneo:
  - O grau do polinômio P =1, 2, 3, 4, 5, 6 e 7.
  - A constante k = 0, 0.25, 0.5, 0.75 e 1. Sendo que a $k = 0$ determina o *kernel* polinomial homogêneo.
- *Kernel Sigmoidal:*
  - κ =0.0001, 0.001, 0.01, 0.1, 1, 10, 100 e 1000.
  - k = 0, -0.25, -0.5, -0.75 e -1.
- *Kernel Gaussiano:*
  - σ =0.0001, 0.001, 0.01, 0.1, 1, 10, 100 e 1000.

Com esses valores é realizado o treinamento, escolhendo entre essas opções, o valor dos parâmetros que determinam a melhor classificação dos dados. Posteriormente, realizam-se os testes.

4.6 VALIDAÇÃO DOS TESTES

A validação é o processo de avaliar como um modelo de classificação é executado em dados reais. Há várias abordagens para avaliar a qualidade e as características de um modelo. Uma avaliação inclui o uso de medidas de validade

estatística para determinar se há problemas nos dados ou no modelo, e outra é separar os dados em conjuntos de treinamentos e testes para testar a exatidão das previsões.

É importante analisar a exatidão e a confiança do modelo. Exatidão é uma medida que mostra se o modelo se relaciona bem com o resultado usando as características extraídas dos dados fornecidos. Na realidade, os valores podem ser ausentes ou aproximados, ou os dados podem ter sido alterados por vários processos. Pode-se decidir aceitar uma determinada quantidade de erros nos dados, especialmente se os mesmos tiverem características bastante uniformes. A confiança avalia o modo como um modelo de dados é executado em conjuntos de dados diferentes. Um modelo é confiável se gerar o mesmo tipo de previsões ou localizar os mesmos tipos gerais de padrões independentemente dos dados de teste fornecidos.

A exatidão e a confiança podem ser analisadas usando a técnica de validação cruzada (*cross-validation*), introduzida pelo estatístico Seymour Geisser. A validação cruzada permite particionar um conjunto de dados em várias seções menores e criar vários modelos nas seções para testar a validade do conjunto inteiro de dados. Uma parte dos dados do conjunto de treinamento é reservada para testes e o restante dos dados é usado para treinamento.

Depois que o conjunto de treinamento foi treinado, o modelo encontrado é usado para fazer previsões no conjunto de testes. Como os dados no conjunto de treinamento são selecionados aleatoriamente nos mesmos dados usados para treinamento, a exatidão derivada dos testes tem menos chance de ser afetada pelas discrepâncias de dados e reflete melhor as características do modelo.

Existem alguns tipos de validação cruzada, que dependem da seleção dos dados para teste:

1. Validação de sub-amostragem aleatórias repetidas: esse método divide aleatoriamente o conjunto de dados em treinamento e validação dos dados. A desvantagem é que pode nunca selecionar uma amostra de pontos, no entanto outra pode ser selecionada várias vezes.
2. Validação cruzada υ*-fold*: nesse método os dados são divididos em υ-subconjuntos. Uma das υ amostras é mantida como conjunto de teste e as $\upsilon - 1$ amostras realizam o treinamento e posteriormente se valida com o conjunto de testes. Esse processo é repetido para as υ amostras. O resultado é a média desses υ testes. Nessa validação são necessárias υ iterações. A vantagem é que todos os dados são utilizados. Caso o conjunto seja desproporcional no número de dados para cada classe,

pode acontecer treinamento do conjunto apenas com dados de uma classe (Diamantidis, Karlis e Giakoumakis, 1998).

3. Validação cruzada "*leave-one-out*": esse método é semelhante ao υ-fold, difere-se apenas no tamanho da amostra, que nesse caso é apenas de um ponto no conjunto de teste. Assim o treinamento é realizado com $l-1$ pontos e o teste realizado com o ponto reservado (Diamantidis, Karlis e Giakoumakis, 1998).
4. *Holdout* validação: o conjunto de teste é escolhido aleatoriamente, cerca de 20 a 30% dos pontos, o restante dos dados são treinados e validados no conjunto reservado (Diamantidis, Karlis e Giakoumakis, 1998).

Esse processo foi utilizado para verificar se os parâmetros que tiveram melhores respostas no treinamento teriam melhores respostas nos testes, ou seja, analisar se os dados de treinamento não eram uma particularidade de alguns pontos.

A validação escolhida foi *Holdout*, por escolher um conjunto aleatoriamente. Esse processo foi realizado várias vezes.

5 EXEMPLOS E VARIAÇÕES DO SVM

Neste capítulo estudam-se algumas aplicações utilizando o SVM e as variações propostas nesse modelo para aperfeiçoamento do mesmo.

5.1 EXEMPLOS E APLICAÇÕES REAIS UTILIZANDO O SVM

Uma técnica de reconhecimento de padrão torna-se importante pela sua utilização em aplicações reais. Essa técnica deve contemplar duas características principais: fácil obtenção de resultados e conter as melhores aproximações. O SVM e suas variações obtiveram bons resultados e destacou-se quando comparado com outras técnicas.

5.1.1 Biometria e segurança

A biometria foi alvo de estudo utilizando a técnica do SVM e de suas variações, conforme exemplificado a seguir.

5.1.1.1 Classificação de faces

Li et al (2000) propuseram o reconhecimento de faces utilizando o *Support Vector Regression* (SVR) com imagens da face com ângulos de inclinação de 10º.

Os dados passam pelo processo de redução de dimensionalidade utilizando Análise de Componentes Principais. Estima-se a posição da cabeça nas imagens. Separam-se as imagens que são cabeças das imagens que não são.

A face humana tem a característica de ser simétrica, em relação ao nariz, em virtude disso, as imagens do lado direito da face são invertidas e tornam-se imagens do lado esquerdo da face, sem perda das características gerais.

Estima-se a posição da cabeça usando SVR, lembrando que as imagens possuem inclinação vertical e horizontal. Posteriormente é realizado o reconhecimento das faces.

Os testes foram realizados obtendo resultados com exatidão acima de 90%.

5.1.1.2 Análise de expressões faciais

Fan et al (2005) aplicaram as técnicas de SVR para reconhecimento de expressões faciais.

Primeiramente, foram detectados a face e os olhos na imagem em tempo real, através do algoritmo de Viola e Jones[31]. Foi realizado um pré-processamento na imagem, posicionando-a no local correto, normalizando seu tamanho, equilibrando a luminosidade e polindo as curvas. O segundo passo foi colocar os dados das características em um vetor. E o terceiro passo, fazer a classificação utilizando SVR. Foram selecionadas seis expressões básicas: sério, ódio, susto, alegre, sorridente e surpreso.

Obteve-se os melhores resultados utilizando o *kernel* gaussiano.

5.1.1.3 Detecção de faces em tempo real em imagem colorida

Palasuthikul et al (2002) aplicaram as técnicas de SVM para detectar faces de imagens coloridas.

O processo consiste em três etapas: na primeira foi realizada a separação dos dados da face e dos dados do plano de fundo, usando-se filtro cor da pele. Depois os dados são analisados para verificar se realmente estão coerentes, usando análise de componentes principais. Finalmente foi utilizado o SVM para reconhecer as faces, em meio a 1000 imagens. Os resultados foram muito bons com alta eficiência e exatidão.

[31]Algoritmo de Viola e Jones é encontrado com mais detalhes em http://www.siliconintelligence.com/people/binu/perception/ (acesso em 02/04/2008)

5.1.1.4 Reconhecimento de assinaturas

Özgündüz et al (2005) propuseram em seu trabalho o reconhecimento de assinaturas, comparando assinaturas originais com imitações.

As assinaturas são compostas de rabiscos e características próprias, como formato das letras, pressão sobre a caneta, velocidade da escrita entre outras, que variam muito de pessoa para pessoa.

O processo proposto possui duas etapas importantes: o treinamento das assinaturas e reconhecimento das assinaturas.

No pré-processamento a assinatura é escaneada, elimina-se o fundo, reduzem-se os ruídos, ajusta-se a um tamanho padrão e elimina-se a diferença de espessura, provocada pela espessura da caneta utilizada.

Algumas características utilizadas no reconhecimento são: informações sobre a densidade da assinatura, altura, maior freqüência horizontal e vertical, número de cantos, entre outras.

Para os resultados desse trabalho, foram utilizadas 1320 assinaturas, originais e imitações. Utilizando SVM com *kernel* gaussiano obtiveram-se ótimos resultados comparados com Redes Neurais Artificiais.

5.1.2 Instituições de crédito

As instituições de crédito podem aproveitar a técnica do SVM para analisar o risco no momento de liberação de crédito a seus clientes.

Lai et al (2005) aplicaram as técnicas de LS-SVM para a análise na avaliação do risco de crédito, comparando com Regressão Linear, Regressão Logística, Redes Neurais Artificiais e SVM.

Para os resultados foi utilizado um conjunto com dados de 1225 clientes, sendo 323 maus credores. As variáveis utilizadas foram 14, destacando-se idade, número de filhos, número de outros dependentes, rendimento, rendimento do cônjuge, valor da residência entre outros.

Primeiramente foi triplicado o número de maus credores, para que o número de bons e maus pagadores fosse igual. Depois se realizou um pré-processamento nos

dados, para que tivessem a média e o desvio padrão com distribuição normal padronizada. Finalmente, treinaram-se e classificaram-se os dados.

Os resultados melhores foram obtidos utilizando a técnica LS-SVM com *kernel* gaussiano.

5.1.3 Medicina e biologia

A busca de novas técnicas de reconhecimento de padrões tem como um dos grandes objetivos melhorar a qualidade de vida. Pode-se detectar doenças, tumores, irregularidades no funcionamento dos órgãos e assim tomar decisões com antecipação, diminuindo o risco de morte e melhorando a qualidade de vida. Portanto, o SVM tornou-se uma ferramenta muito interessante.

5.1.3.1 Classificação de arritmia

Joshi et al (2005) propuseram um reconhecimento de arritmia utilizando os expoentes de Hölder Local e *Support Vector Machine*.

Arritmia é um termo comum para ritmos cardíacos com irregularidades e desigualdades em relação aos sinais normais. A classificação e caracterização desses sinais são importantes para monitorar o coração dos indivíduos.

Primeiramente, os sinais foram pré-processados para eliminação de outros ruídos. Posteriormente, foram computados os Coeficientes de Wavelet[32] para selecionar as escalas e utilizá-las para calcular os expoentes de Hölder local, selecionar as características e classificar através do SVM.

Para os resultados foram utilizados dados de aproximadamente 500 ritmos cardíacos normais e também com arritmia. Foi realizada classificação binária (classe normal e classe de arritmia), depois foi realizada classificação múltipla (tipos de arritmia: Fibrilação Atrial, Ventricular e Nodal). Ambas as classificações obtiveram grande exatidão nos resultados.

[32] Wavelets são ondas pequenas (*ondeletes*, em francês) com determinadas propriedades que as tornam adequadas a servirem de base para decomposição de outras funções, assim como senos e cossenos servem de base para decomposições de Fourier.

5.1.3.2 Diagnóstico de diabetes mellitus

Stoean (2005) aplicou as técnicas de SVM associadas a algoritmos evolucionários (genéticos), o *Evolutionary Support Vector Machines* (ESVM), no diagnóstico da doença *diabetes mellitus*.

Os dados coletados eram de pacientes do sexo feminino, com no mínimo 21 anos, de uma cidade próxima de Phoenix, no Arizona, EUA. As características escolhidas eram oito, entre elas, a idade, o número de gestações, a pressão arterial, o índice de massa corporal, entre outros dados médicos.

Foram utilizados 768 dados, sendo que 34,9% possuem diabetes *mellitus*. Para encontrar os melhores parâmetros foi realizada a validação cruzada. Comparado com outras técnicas, obteve ótimos resultados de acertos.

5.1.3.3 Diagnóstico de síndrome genética

David e Lerner (2005) aplicaram as técnicas de SVM no diagnóstico de síndromes genéticas, ou seja, possíveis anormalidades encontradas nos cromossomos[33].

Os dados, sinais FISH[34], foram normalizados e linearizados. O SVM com *kernel* gaussiano sobressaiu-se sobre os *kernel* linear e polinomial.

Comparando-se os resultados de SVM gaussiano com outras técnicas de reconhecimento de padrão, obtiveram-se ótimos resultados.

5.1.4 Indústria

As indústrias podem obter muitas vantagens utilizando as técnicas de reconhecimento de padrões, evitando desperdícios e obtendo melhores lucros. O SVM é uma das técnicas que vem se destacando nesse ramo.

---

[33] Por exemplo, Síndrome de Down ou mongolismo, que tem um cromossomo a mais no par 21 do DNA. Entre outras características, o indivíduo que possui esta síndrome apresenta deficiência mental, inflamação das pálpebras e prega única no dedo mínimo dos pés.

[34] FISH = *Fluorescence In Situ Hybridization.*

5.1.4.1 Previsão de demanda de energia elétrica

Ruas et al (2008) aplicaram as técnicas de SVR e Redes Neurais Artificiais na previsão da demanda de energia elétrica.

Devido à privatização do setor de energia elétrica e a comercialização de energia entre distribuidoras, torna-se importante para estas empresas à previsão de demanda tanto em longo quanto em curto prazo.

A previsão da demanda em longo prazo é necessária para contratação de energia a ser comprada e a de curto prazo para garantir que a distribuição trabalhe dentro dos limites contratados. Foram analisados os dados, considerando a sazonalidade diária e semanal.

Os resultados foram satisfatórios comparados à literatura. Uma vantagem é que o SVR requer apenas um único parâmetro a ser ajustado, ao contrario das redes neurais, a qual precisa ser definida a sua estrutura.

5.1.4.2 Quantificar os adulterantes no leite em pó

Ferrão et al (2007) aplicaram técnicas de LS-SVM para analisar as quantidades de adulterantes no leite em pó.

Com o aumento na exportação de leite em pó, aumentaram as exigências na qualidade do leite. As fraudes são freqüentes, com acréscimo de soro de leite, maltose, sacarose, amido e até soda cáustica.

O processo para análise de adulterantes no leite incluiu a contaminação das amostras de leite em pó com amido, sacarose e com soro. Definiram-se os parâmetros sigma, gama e as matrizes dos espectros para a calibração e precisão do modelo.

Foram comparadas as técnicas de regressão por mínimos quadrados parciais e LS-SVM, a qual sobressaiu nos resultados e na capacidade de generalização.

5.2 VARIAÇÕES DO SVM

A formulação original do SVM, descrita anteriormente, com a utilização da norma-2, do vetor de pesos w e das variáveis de folga ξ, não é a única maneira de expressar

matematicamente a maximização da margem. Várias modificações foram propostas na literatura.

5.2.1 *Evolutionary Support Vector Machines*

Stoean (2005) propôs uma técnica para classificação de conjuntos usando o SVM e algoritmos evolucionários (genéticos).

Para conjuntos linearmente separáveis, representam-se os cromossomos como vetores, cujas coordenadas são os pesos w e o *bias* b, da forma: $c = (w_1, w_2, ..., w_n, b)$, sendo que as coordenadas estão normalizadas, ou seja, $w_i \in [-1, +1]$ e $b \in [-1, +1]$. Na população inicial, cada gene é gerado aleatoriamente, com coordenadas no intervalo $[-1, +1]$.

A Função *Fitness* avaliada é dada por:

$$f(c) = f(w_1, ..., w_n, b) = w_1^2 + ... + w_n^2 + \sum_{i=1}^{n} \left[t\left(y_i \left(w^t x - b \right) - 1 \right) \right]^2 \tag{3.49}$$

onde:

$$t(a) = \begin{cases} a, a < 0 \\ 0, a \geq 0 \end{cases}$$

E as variações de operações são os cruzamentos, ou *crossover*, e mutações com perturbação normal. Na função *Fitness* foi adicionada a variável de folga ξ para obter uma função para dados não lineares.

O critério de parada é o número de gerações fornecido no algoritmo. O objetivo é encontrar o mínimo $(f(c), c)$, para identificação da presença de *diabetes mellitus*. Os resultados foram ótimos comparados com outras técnicas, como o CPLEX, software comercial utilizado para resolução de problemas quadráticos, e algumas variações do SVM, como o SVMlight, proposto por Joachims em 1999, o ASVM, Active Support Vector Machine proposto por Mangasarian e Musicant em 2000 e o CSVM, Critical Support Vector Machine proposto por Raicharoen e Lursinsap em 2002.

5.2.2 *Generalized Support Vector Machine*

Mangasarian (2001) propôs o *Generalized Support Vector Machine* (GSVM), que utiliza a formulação original e acresce um termo extra na função objetivo primal. Assim o SVM passa a ser um caso particular do GSVM, quando o termo extra é nulo.

Pode-se analisar que o produto das matrizes $K(A,A')K(A,A')'$ resulta em uma matriz semi-definida positiva. Necessariamente, a função *kernel* K não é simétrica, nem semi-definida positiva, nem precisa ser contínua, ou seja, não é um produto interno (MANGASARIAN, 2001).

Portanto, o GSVM é capaz de utilizar funções *kernel* mais gerais, sendo que não necessitam satisfazer o teorema de Mercer inteiramente, gerando soluções que não seriam possíveis com o SVM (CARVALHO, 2005).

5.2.3 *Smooth Support Vector Machine*

O modelo *Smooth Support Vector Machine* (SSVM) é uma reformulação do SVM para problema irrestrito (CARVALHO, 2005).

Para a resolução do SSVM é utilizado o algoritmo de Newton-Armijo para SSVM. A diferença entre o SSVM e o SVM, é que o primeiro é um sistema linear, e o segundo é um problema quadrático. Além disso, o SSVM possui convergência quadrática (MANGASARIAN, 2001).

5.2.4 *Lagrangian Support Vector Machine*

A técnica do *Lagrangian Support Vector Machine* (LSVM) abordada por Mangasarian (2001) é utilizada quando o conjunto de padrões envolve milhões de pontos a serem classificados.

O algoritmo do LSVM é baseado nas condições de KKT e o resultado é obtido de forma iterativa, através de uma fórmula de recorrência. Este algoritmo, LSVM, possui convergência linear global.

5.2.5 *Reduced Support Vector Machine*

O objetivo do *Reduced Support Vector Machine* (RSVM) é generalizar a superfície de separação de grandes bancos de dados utilizando apenas um subconjunto de dados para caracterização, escolhidos aleatoriamente. Com a redução de alguns pontos, a matriz K dos *kernel* é retangular.

Lee e Mangasarian (2001) descrevem com detalhes o algoritmo RSVM. A qualidade dos resultados do RSVM é mantida, a vantagem em relação ao SVM clássico é o tempo computacional e a memória utilizada reduzidos.

5.2.6 *Least Squares Support Vector Machine*

A formulação do *Least Squares Support Vector Machine* (LS-SVM) possui algumas modificações no modelo primal, possibilitando a resolução através de sistemas de equações lineares. Aplicando a formulação dual, através das condições de KKT, obtem-se uma restrição que difere das equações do SVM: $\alpha_i = C\xi_i$. Em virtude disso, os Multiplicadores de Lagrange α_i serão proporcionais aos erros ξ_i (CARVALHO e BRAGA, 2004).

Uma característica marcante do SVM é o fato da maioria dos multiplicadores de Lagrange, associados aos vetores de treinamento, serem nulos. O mesmo não ocorre no LS-SVM, seus valores se distribuem, sem predominância de valores nulos (CARVALHO e BRAGA, 2004).

A vantagem do LS-SVM é apresentar um sistema linear, o qual possui vantagens na resolução, comparando com o SVM dual, que é um problema quadrático.

5.2.7 ν - *Support Vector Machine*

Chen, Lin & Scholkopf (2005) formularam uma alteração no SVM, eliminando o parâmetro C, e introduzindo o parâmetro $\nu \in [0,1]$, o qual possibilita grau de liberdade. O objetivo do ν - *Support Vector Machine* (ν-SVM) é reduzir a envoltória convexa,

aumentando a margem de distância entre os conjuntos (CHEN, LIN e SCHOLKOPF, 2005).

Para a obtenção dos resultados, essa técnica utiliza o Método da Decomposição, por exemplo, o SMO, com algumas alterações (CHEN, LIN e SCHOLKOPF, 2005).

5.2.8 *Proximal Support Vector Machine*

O *Proximal Support Vector Machine* (PSVM) não classifica os pontos através de uma superfície de separação, ao contrário, a classificação é feita pela proximidade dos pontos em relação a dois hiperplanos paralelos, que são dispostos o mais longe possível um do outro (CARVALHO, 2005).

5.2.9 *Semi-Supervised Support Vector Machine*

O *Semi-Supervised Support Vector Machine* (S³VM) consiste na realização de um treinamento semi-supervisionado, ou seja, apenas uma parte das entradas tem saídas associadas. Caso não se conheça nenhuma saída, o treinamento se torna não supervisionado, podendo ser utilizada alguma técnica como *clustering*[35].

O S³VM foi proposto para melhoria de problemas nos quais nem todos os dados possuem saídas conhecidas y (CARVALHO, 2005).

5.2.10 Classificação múltipla utilizando o SVM

O SVM foi inicialmente desenvolvido para classificação binária e recentemente tem sido desenvolvida uma extensão do SVM para várias classes. Muitas abordagens para múltiplas classes do SVM decompõem o conjunto de dados em vários problemas binários. Por exemplo, o método um-contra-um, "*one-against-one*", treina o SVM binário para duas classes de dados quaisquer e obtém uma função de decisão. E assim

[35] Clustering é um termo inglês, que significa agrupar. Existe também uma técnica estatística chamada de Clusterização que tem a finalidade de agrupar dados em conjuntos.

sucessivamente para as k classes. No final, obtem-se $k(k-1)/2$ funções de decisão. Utiliza-se uma estratégia de votação para designar novos padrões a suas respectivas classes.

Chen, Lin e Scholkopf (2005) estenderam o conceito de múltipla classificação para o ν-SVM com *kernel* gaussiano. Primeiramente, utilizaram o método um-contra-um para cada duas classes e obtiveram os melhores parâmetros (C,σ), próprios para as duas classes treinadas. Então foi pré-selecionado (C,σ) e treinado para selecionar o melhor modelo. Assim, cada função decisão possui o mesmo (C,σ). Isso pode não ser muito bom para as $k(k-1)/2$ funções de decisão, porém evita problemas de *overfitting*. Esse processo teve grande êxito nos testes realizados por Chen, Lin e Scholkopf (2005).

As aplicações vistas no capítulo são as mais variadas e utilizam algumas variações do SVM. As variações propostas estão limitadas a um determinado conjunto de dados. Escolheu-se trabalhar com o SMO por trabalhar com apenas dois pontos em cada iteração, diminuindo o tempo computacional.

6 ANÁLISE DE RESULTADOS

Foram realizados os treinamentos para a escolha da função *kernel* e de seus parâmetros mais adequados a cada base de dados. Encontrados os melhores parâmetros foram realizados os testes e a validação cruzada, para avaliar se os parâmetros utilizados eram ideais para cada banco de dados.

Os treinamentos foram executados através do *software* implementado e alguns parâmetros, que tiveram melhor desempenho, encontram-se nas tabelas deste capítulo. No apêndice encontram-se as tabelas com mais parâmetros.

Como critério de escolha dos melhores parâmetros utilizou-se o fato de possuir menor porcentagem de erros através do treinamento realizado, o número de vetores suportes ser uma fração do conjunto de dados e o valor do parâmetro C, que pondera o valor dos erros, quanto menor o seu valor, maior a distância entre os conjuntos, ou seja, a superfície está mais generalizada.

Estabeleceu-se como notação de vetor suporte não *bound* ($0 < \alpha < C$) a sigla VS, o vetor suporte *bound* ($\alpha = C$) como VS-*bound* e validação cruzada como VC.

6.1 DADOS ICTERÍCIA SEM TRATAMENTO

Os dados de icterícia foram treinados utilizando os *kernel* e parâmetros escolhidos e variando o valor do parâmetro C.

Utilizando o *kernel* sigmoidal, o treinamento obteve erros abaixo de 3,5% e o número de vetores suportes se mantém constante em 70, sendo todos *bound*. Com o *kernel* gaussiano também não ocorre erros no treinamento, porém, todos os dados são vetores suportes.

Quando se utiliza o *kernel* polinomial os erros são elevados, acima de 16%. No entanto, o número de vetores suportes é baixo. Os parâmetros que melhor se destacaram foram $C = 0.01$, $p = 1$ e $k = 0.75$, com erro de 16,95% e 52 vetores suportes, sendo 25 *bounds*.

Conclui-se que, com os dados sem tratamento, as funções de decisão não são adequadas para a classificação, devido a porcentagem elevada de erros, este problema pode estar relacionado com as diferenças de unidades entre as características.

As tabelas com os parâmetros e seus respectivos erros de treinamento, estão detalhadas no apêndice 2.

6.2 DADOS ICTERÍCIA NORMALIZADOS

Os dados de icterícia foram normalizados, seus valores possuíam grandezas diferentes, assim os valores permaneceram no intervalo $[-1,1]$. Foram treinados através do SVM e os melhores parâmetros encontrados em relação ao número de erros, número de vetores suportes e vetores suportes *bounds* estão descritos na tabela a seguir:

TABELA 8 – MELHORES PARÂMETROS DO SVM PARA O PROBLEMA ICTERÍCIA

KERNEL	PARÂMETROS	C	Nº VS	Nº VS *BOUNDS*	Nº ERROS	ERRO TREINO (%)
Polinomial	P=5 e k=0.5	0.1	31	15	5	4,24
Polinomial	P=6 e k=0.25	0.1	33	9	2	1,70
Polinomial	P=6 e k=0.25	1	38	1	2	1,70
Polinomial	P=5	10	40	1	1	0,85
Polinomial	P=6 e k=1	1	37	1	0	0,00
Polinomial	P=5 e k=0.25	10	38	1	0	0.00
Polinomial	P=5 e k=0.5	10	37	0	0	0.00
Polinomial	P=5 e k=1	10	37	0	0	0.00
Polinomial	P=6 e k=0.25	10	36	0	0	0.00
Polinomial	P=6 e k=0.5	10	37	0	0	0.00
Sigmoidal	Kapa=1	10	0	70	0	0.00

FONTE: A autora (2008)

Os resultados de todos os treinamentos realizados com os dados normalizados encontram-se no apêndice 2.

A função *kernel* polinomial não homogêneo obteve os melhores resultados em relação com os demais *kernel*, principalmente os de grau 5 e 6. Para verificar o desempenho do *kernel* gaussiano de parâmetros $p = 5$, $K = 1$ e $C = 10$, utilizou-se a validação cruzada, obtendo-se bons resultados como pode ser visto na tabela a 9:

TABELA 9 – VALIDAÇÃO NO PROBLEMA ICTERÍCIA UTILIZANDO *KERNEL* POLINOMIAL NÃO HOMOGÊNEO

Nº PONTOS TREINO	Nº PONTOS TESTE	ERRO TREINO (*)	ERRO TESTES (*)	PONTOS TESTE	Nº VS (*)	Nº VC
92	26	0,89%	8,15%	20%	30	200
99	19	1,40%	7,71%	15%	34	200
105	13	1,89%	7,50%	10%	32	200
110	8	1,92%	5,10%	5%	34	200

(*) Em média.

FONTE: A autora (2008)

O número de vetores suportes é aproximadamente 32% dos dados de treinamento e isso gerou uma superfície de separação bem generalizada.

A função *kernel* sigmoidal também obteve bons resultados, porém todos os seus vetores suportes são *bounds*. Utilizando a validação cruzada para o *kernel* sigmoidal com $\kappa = 10$ e $C = 10$, os resultados podem ser vistos na tabela 10:

TABELA 10 – VALIDAÇÃO NO PROBLEMA ICTERICIA UTILIZANDO *KERNEL* SIGMOIDAL

Nº PONTOS TREINO	Nº PONTOS TESTE	ERRO TREIN.(*)	ERRO TESTES (*)	PONTOS TESTE	Nº VS (*)	Nº VC
92	26	0,00%	20,81%	20%	54	200
99	19	0,00%	14,95%	15%	58	100
99	19	0,00%	11,27%	10%	64	200
110	8	0,00%	2,25%	5%	62	200

(*) Em média.

FONTE: A autora (2008)

Pode ser verificado que a função sigmoidal não possui erros de treinamentos, no entanto isso não gera boas superfícies de separação, pois a média de erros de teste é elevada. Isso pode estar ocorrendo pelo fato de que aproximadamente 60% dos dados de treinamento são vetores suporte, além disso, todos os vetores suportes são *bounds* nos treinamentos realizados.

Quanto ao kernel polinomial não homogêneo de grau 5 e constante 1, este generaliza a superfície de separação, sendo que durante o treinamento obteve uma média de erros baixa. Durante os testes, utilizando 20% dos dados para o teste, obtém-se um bom desempenho com média de erro de 8,15%, resultado melhor do que os relatados por Steiner (1995).

6.3 DADOS *ADULT*

Os dados *Adult* possuem uma separação em banco de dados menores. Todos os sub-bancos possuem o banco de dados para treinamento e um banco para testes. O banco de dados de treinamento A1A, que possui 1605 dados, foi treinado com vários parâmetros. Alguns parâmetros, que tiveram destaque no treinamento, podem ser analisados na tabela 11:

TABELA 11 – MELHORES PARÂMETROS DO SVM PARA A1A

KERNEL	PARÂMETROS	C	Nº VS	Nº VS *BOUNDS*	Nº ERROS	ERRO TREINO (%)
Sigmoidal	κ =10 e k=-0.75	10	0	790	2	0,13
Gaussiano	σ =0,01	1	310	729	14	0,87
Polinomial	P=5 e k=0.5	0,1	655	2	20	1,25
Polinomial	P=6 e k=0	1	759	0	22	1,37
Gaussiano	σ =1	1	556	679	22	1,37
Polinomial	P=3 e k=0	0,01	529	59	40	2,49
Polinomial	P=4 e k=0	0,0001	442	200	60	3,74
Polinomial	P=2 e k=0,75	1	412	55	91	5,67

FONTE: A autora (2008)

A tabela com os parâmetros que obtiveram erros inferiores a 10% encontra-se no apêndice 4.

Testando o banco de teste A1At, contendo 30.956 pontos, obtiveram-se os seguintes resultados, apresentados na tabela 12:

TABELA 12 – CLASSIFICAÇÃO PARA A1AT

KERNEL	PARÂMETROS	C	Nº ERROS	ERRO (%)
Sigmoidal	κ =10 e k=-0.75	10	100	0,32
Gaussiano	σ =0,01	1	7398	23,89
Polinomial	P=5 e k=0.5	0,1	6357	20,53
Polinomial	P=6 e k=0	1	8066	26,05
Gaussiano	σ =1	1	7391	23,87
Polinomial	P=3 e k=0	0,01	6608	21,34
Polinomial	P=4 e k=0	0,0001	6498	20,99
Polinomial	P=2 e k=0,75	1	7442	24,04

FONTE: A autora (2008)

Conclui-se que o *kernel* sigmoidal obteve o melhor resultado, mesmo com todos os vetores suportes *bound*. No entanto, quando se utilizou o *kernel* polinomial a porcentagem de erros nos pontos de teste foi superior a 20%.

O insucesso dos parâmetros treinados pode estar relacionado ao número de dados e de características. O banco de dados de treinamento possui 1605 dados, no entanto o banco de dados de teste possui 30.956 dados.

6.4 DADOS *GERMAN*

O banco de dados *German* possui 999 dados com 24 características. Esses dados possuem valores normalizados entre $[-1,1]$ e outros com resposta dicotômica, -1 ou 1. No treinamento alguns dados se destacaram, como pode ser visto na tabela 13:

TABELA 13 – MELHORES PARÂMETROS DE TREINAMENTO PARA *GERMAN*

KERNEL	PARÂMETROS	C	Nº VS	Nº VS *BOUNDS*	Nº ERROS	ERRO TREINO (%)
Gaussiano	σ =0,01	1	61	591	0	0,00
Gaussiano	σ =0,1	10	999	0	0	0,00
Polinomial	P=4 e k=0	0,001	504	1	10	1,00
Polinomial	P=3 e k=1	1	445	0	23	2,30
Sigmoidal	κ =0,01 e k=0,75	100	41	448	178	17,82

FONTE: A autora (2008)

O kernel gaussiano não possui erros no treinamento, no entanto, todos os dados são vetores suporte quando $\sigma = 0,1$ e $C = 10$. O kernel polinomial possui erros no treinamento, mas não utiliza todos os dados como vetores suportes. O *kernel* sigmoidal não obteve um bom desempenho no treinamento.

Esse banco de dados não possui um banco de testes, então foi realizada a validação cruzada para alguns parâmetros, que obtiveram melhor desempenho no treinamento, como pode ser verificado na tabela 14:

TABELA 14 – VALIDAÇÃO NO BANCO DE DADOS *GERMAN*

KERNEL	PARÂMETROS	C	PONTOS TESTE	ERRO TREINO (*)	ERRO TESTES (*)	Nº VS (*)	Nº VC
Gaussiano	σ =0,01	1	20%	0,00%	5,94%	707	50
Gaussiano	σ =0,1	10	20%	1,66%	8,75%	422	50
Polinomial	P=4 e k=0	0,001	20%	1,61%	7,86%	422	50
Polinomial	P=3 e k=1	1	20%	3,29%	9,58%	359	50

(*) Em média.

FONTE: A autora (2008)

Verifica-se que os testes com o *kernel* gaussiano e o polinomial formam boas superfícies de separação. A diferença está no parâmetro C.

6.5 DADOS *AUSTRALIAN*

O banco de dados *Autralian* contém 690 pontos com 14 características, sendo que os valores estão normalizados entre -1 e 1. O treinamento dos dados teve destaque para os parâmetros apresentados na tabela 15:

TABELA 15 – MELHORES PARÂMETROS DE TREINAMENTO PARA *AUSTRALIAN*

KERNEL	PARÂMETROS	C	Nº VS	Nº VS *BOUNDS*	Nº ERROS	ERRO (%)
Gaussiano	σ =0,001	1	43	604	0	0,00
Gaussiano	σ =0,01	1	263	423	2	0,29
Gaussiano	σ =0,1	10	520	16	24	3,48
Polinomial	P=6 e k=0,75	0,0001	209	16	39	5,65
Polinomial	P=5 e k=1	0,001	162	42	45	6,52
Polinomial	P=3 e k=1	0,1	118	77	49	7,10
Sigmoidal	κ =0,1 e k=1	10	23	178	80	11,59

FONTE: A autora (2008)

Esse banco de dados não possui dados de teste, portanto realizou-se a validação cruzada com os dados, como se verifica na tabela 16:

TABELA 16 – VALIDAÇÃO NO BANCO DE DADOS *AUSTRALIAN*

KERNEL	PARÂMETROS	C	PONTOS TESTE	ERRO TREINO (*)	ERRO TESTES (*)	Nº VS (*)	Nº VC
Gaussiano	σ =0,001	1	20%	0,00%	8,71%	539	20
Gaussiano	σ =0,01	1	20%	0,15%	8,17%	547	50
Gaussiano	σ =0,1	10	20%	6,45%	8,56%	404	50
Polinomial	P=6 e k=0,75	0,0001	20%	8,35%	8,75%	160	20
Polinomial	P=5 e k=1	0,001	20%	9,32%	13,25%	142	20
Polinomial	P=3 e k=1	0,1	20%	6,80%	7,39%	157	20

(*) Em média.

FONTE: A autora (2008)

Os dados *Australian* obtiveram bom desempenho quando se utilizou o *kernel* gaussiano para o treinamento. No entanto, o kernel polinomial de grau 3 obteve melhor desempenho, pois utilizou menor número de vetores suportes e generalizou melhor, com erro de treinamento de 6,80% e nos testes 7,39%.

No próximo capítulo estão as conclusões sobre os testes realizados.

CONCLUSÕES E SUGESTÕES DE TRABALHOS FUTUROS

Neste capítulo apresentam-se as conclusões desta dissertação bem como as sugestões para trabalhos que futuramente podem ser desenvolvidos utilizando a técnica do SVM e também suas variações.

6.6 CONCLUSÕES

O SVM é uma técnica de reconhecimento de padrões muito robusta e gera boa generalização nos resultados (Cristianini e Shawe-Taylor, 2000)

O modelo primal do SVM separa classes linearmente separáveis sendo aplicável apenas a poucos problemas reais. A inclusão de variáveis de folga no problema primal permite a classificação de pontos incorretamente, aceitando alguns erros de classificação que podem ter ocorrido na coleta de dados. A importância dos erros está relacionada ao parâmetro C, quanto menor seu valor, maior é a margem de separação entre as classes e em virtude disso existirão pontos classificados erroneamente. Quanto maior o valor do parâmetro C, mais pontos serão classificados corretamente, no entanto a margem de separação será menor.

Devido às dificuldades geradas pelas restrições do modelo primal do SVM utiliza-se a teoria do Lagrangeano para obter a formulação dual, na qual se diminui o número de restrições.

No modelo dual aparece um produto interno, que pode ser substituído por uma função *kernel*, desde que essa função obedeça às condições de Mercer. Com a função *kernel* é possível separar dados que não são linearmente separáveis. Porém, além de se ter que escolher a função *kernel* adequada a cada situação, surge outra dificuldade que é a de calibrar os pesos da mesma, afim de que as superfícies separadoras sejam generalizadas.

A formulação do SVM envolve um problema quadrático convexo, que possui solução única, no entanto requer ferramentas complexas para sua resolução. Para isso foi proposta a utilização do algoritmo SMO e sua implementação, para agilizar o processo de resolução.

Os treinamentos e testes foram realizados usando as bases de dados Icterícia, *Adult*, *German* e *Australian*, onde apenas os dados *Adult* possui um conjunto de teste separado.

No treinamento foram utilizados vários valores do parâmetro C para as funções *kernel* polinomial, sigmoidal e gaussiano, com vários parâmetros, para a escolha dos melhores, ou seja, aqueles que apresentam menores erros de treinamento e não utilizam todos os dados como vetores suportes.

Para os bancos de dados que não possuíam banco de testes foi realizada a validação cruzada *holdout*, na qual 20% dos dados de treinamento eram separados para realização de testes. Esse percentual de dados foi escolhido aleatoriamente, tomando o cuidado de manter a proporção de dados das duas classes.

Nos dados Icterícia normalizados, uma das funções *kernel* que se destacou devido ao bom desempenho e baixo número de vetores suportes foi o *kernel* polinomial não homogêneo de grau $p = 5$, $k = 1$ e $C = 10$. Com esses parâmetros obtiveram-se erro de treinamento de 0,89% e nos testes 8,15% e apenas 30 vetores suportes, durante a validação cruzada. O resultado é muito bom comparado com os relatados anteriormente publicados.

Nos dados *Adult,* o *kernel* sigmoidal teve destaque utilizando os parâmetros $\kappa = 10$, $k = -0,75$ e $C = 10$. Durante o treinamento dos 1.605 dados apresentou erro de 0,13% e utilizando a superfície separadora gerada por esses dados no banco de dados de teste contendo 30.956 dados obteve-se erro de teste de 0,32%, ou seja, apenas 100 dados ficaram classificados incorretamente. O número de vetores suportes utilizados nessa superfície separadora foi de 790. Considerando o tamanho do conjunto de treinamento com o conjunto de teste, obteve-se uma ótima superfície separadora.

Nos dados *German*, o *kernel* gaussiano obteve destaque quando os parâmetros utilizados são $\sigma = 0,01$ e $C = 1$. Durante o treinamento não se obtiveram erros e nos testes o erro médio foi de 5,94%. Este resultado é considerado bom, porém o número de vetores suportes é de aproximadamente 88%. Realizando os testes para a mesma função *kernel* e alterando os parâmetros $\sigma = 0,1$ e $C = 10$ o erro de treinamento foi de 1,66% e no teste foi de 8,75%, sendo utilizado como vetor suporte 422 dados, 50% dos dados de treinamento. O mesmo número de vetores suporte é encontrado quando se utiliza a função *kernel* polinomial de grau 4 e $C = 0,001$, com erro de treinamento de aproximadamente 1,66% e de teste de 7,86%.

Nos dados *Australian*, o *kernel* gaussiano obteve destaque com os parâmetros $\sigma = 0,001$ e $C = 1$ não obtendo erro de treinamento e com erro de teste de 8,71%, utilizando todos os dados como vetores suporte. Utilizando o kernel polinomial de grau 3, $k = 1$ e $C = 0,1$ o erro de treinamento foi maior, com cerca de 6,80%. No entanto, no teste o erro foi reduzido para o valor de 7,39%, utilizando 157 vetores suportes, com cerca de 28% dos dados de treinamento.

Com os treinamentos e teste realizados usando o algoritmo SMO obtiveram-se superfícies de separação generalizadas e robustas. O SMO é um algoritmo que possibilitou a resolução do problema de SVM com apenas dois pontos otimizados em cada iteração, evitando problemas computacionais.

Conclui-se que para obter bons resultados no reconhecimento de padrões utilizando o SVM é necessário ter um conjunto de dados representativo para aplicação, ou seja, não seja uma particularidade; os dados contendo informações com grandezas semelhantes, ou seja, normalizadas; e escolhendo uma função *kernel* adequada à aplicação, juntamente com o valor dos parâmetros;

Os primeiros dois critérios descritos acima são necessários para outras técnicas de reconhecimento de padrões. A escolha do *kernel* é uma particularidade do SVM, assim como definir a arquitetura de uma rede neural é para RNA.

O SVM se sobressai a outras técnicas em relação a dados coletados incorretamente, pois em sua formulação permite dados classificados incorretamente, ou seja, é uma ferramenta robusta (Cristianini e Shawe-Taylor, 2000). A análise do número de vetores suportes permite verificar se os parâmetros escolhidos tiveram um bom desempenho no treinamento dos dados, ou seja, obteve-se generalização do processo de classificação de padrões.

Portanto, o SVM é uma ótima ferramenta de reconhecimento de padrões sendo resolvido através do algoritmo SMO, proporcionando resultados robustos e generalizados.

6.7 SUGESTÕES DE TRABALHOS FUTUROS

A seguir estão descritas algumas sugestões que poderão ser objeto de estudos futuros de modo a aperfeiçoar a metodologia aqui abordada:

- Implementar essa técnica de reconhecimento de padrões para maior número de classes;
- Utilizar o SVM como ferramenta para regressão;
- Aprofundamento teórico sobre as funções *kernel*;
- Melhorias do algoritmo como, por exemplo, utilizar três ou mais pontos em cada iteração do SMO.

REFERÊNCIAS

BISHOP, C. M. ***Neural Networks for Pattern Recognition***. New York: Oxford University Press, 1995.

BITTENCOURT, H. R. **Reconhecimento Estatístico de Padrões: O Caso da Discriminação Logística Aplicada à Classificação de Imagens Digitais Obtidas por Sensores Remotos.** In: Congresso Brasileiro de Computação, 1., 2001, Itajaí-SC. Anais... Santa Catarina: 2001, p. 485-493.

BURDEN, R. L.; FAIRES, J. D. **Análise Numérica**. . São Paulo: Pioneira Thomson Learning, 2003.

CARVALHO, B. P. R.; BRAGA, A. P. **Estratégias neurais para treinamento de Least Squares Support Vector Machines**. In:Brazilian Symposium on Artificial Neural Networks, VIII., 2004, São Luiz, Maranhão.

CARVALHO, B. P. R. **Novas Estratégias para Detecção Automática de Vetores de Suporte em *Least Squares Support Vector Machines*.** Dissertação (Mestrado em Engenharia Elétrica) – Universidade Federal de Minas Gerais, MG, 2005.

CARVALHO, B. P. R. **O Estado da Arte em Métodos de Reconhecimento de Padrões: *Support Vector Machines*.** In: Congresso Nacional de Tecnologia e Comunicação, 2005, Belo Horizonte, Minas Gerais.

CHEN, P. H.; LIN, C. J. SCHOLKOPF, B. **A Tutorial on v-Support Vector Machine**. Appl Stochastic Models Business Industry 2005; 21:111–136.

CHANG, C. LIN, C. **Libsvm: a library for support vector machines**. Disponível online: http://www.csie.ntu.edu.tw/_cjlin/libsvm, 2001. Acesso em fevereiro/2007

CORREA, E. S. **Algoritmos Genéticos e Busca Tabu Aplicados ao Problema das P-Medianas**. Dissertação (Mestrado em Métodos Numéricos para Engenharia) – Universidade Federal do Paraná, PR, 2000.

CORTES, C. VAPNIK, V. ***Support-Vector Networks***. Machine Learning, Volume 20, Número 3, Setembro/1995, pg 273-297, Editora Springer,1995.

CRISTIANINI, N.; SHAWE-TAYLOR, J. ***An Introduction to Support Vector Machine and Other Kernel-Based Learning Methods***. United Kingdom: Cambridge, 2000.

CUNICO, L. H. B. **Técnicas em Data Mining Aplicadas na Predição de satisfação de Funcionários de uma Rede de Lojas do Comércio Varejista**. Dissertação (Mestrado em Métodos Numéricos para Engenharia) - Universidade Federal do Paraná, PR, 2005.

DAVID, A. LERNER, B. ***Support Vector Machine-Based image classification for genetic syndrome diagnosis***. Pattern Recognition Letters, volume 26, número 8, pg 1029-1038. Junho, 2005

DIAMANTIDIS, N.A. KARLIS, D. GIAKOUMAKIS, E. A. **Unsupervised stratification of cross-validation for accuracy estimation.** Artificial Intelligence, UK, volume 116, p. 1-16. 1-2 (Jan. 2000).

EHRICH, P. J. **Pesquisa Operacional: Curso Preparatório**. Atlas, 3ªEdição. São Paulo, 1980.

FAN, C.; SARRAFZADEH, H.; DADGOSTAR, F.; GHOLAMHOSSEINI, H. ***Facial Expression Analysis by Support Vector Regression***. Proceedings of Image and Vision Computing, Dunedin, New Zealand, 2005

FAYYAD, U.M., PIATETSKY-SHAPIRO, G., SMYTH, P., UTHRUSAMY, R. **Advances in knowledge Discovery & Data Mining**. AAAI/MIT, 1996.

FERNANDES, C. **Comparação Entre o Índice de Integridade Biótica e um Método de Multicritério, para Análise da Qualidade Ambiental de Três Riachos Tributários ao Reservatório de Itaipu***.* Dissertação (Mestrado em Métodos Numéricos em Engenharia) – Universidade Federal do Paraná, PR, 2006.

FERRÃO, M. F.; MELLO, C.; BORIM, A.; MARETTO, D. A.; POPPI, R. J. **LS-SVM: uma nova ferramenta quimiométrica para regressão multivariada. Comparação de modelos de regressão LS-SVM e PLS na quantificação de adulterantes em leite em pó empregando NIR**. Revista Química Nova. Volume nº 30. Edição nº 4. São Paulo. Julho/Agosto 2007.

FRIEDLANDER, A. **Elementos de Programação Não-Linear**. São Paulo: Editora Unicamp, Série Manuais, 1994.

GROBE, J. R. **Aplicações da Estatística Multivariada na Análise de Resultados em Experimentos com Solo e Animais**. Dissertação (Mestrado em Métodos Numéricos) – Universidade Federal do Paraná, PR, 2005.

GUIMARÃES, I. A.; CHAVES NETO, A. **Reconhecimento de Padrões: Comparação de Métodos Multivariados e Redes Neurais**. Revista Negócios e Tecnologia da Informação. Curitiba, Vol. 1, No 1, pg 38-58, 2006.

HAIR JR, J. F.; ANDERSON, R. E.; TATHAM, R. L.; BLACK, W. C. ***Multivariate Data Analysis***. Prentice Hall. Upper Saddler River, New Jersey. Fifth Edition. 1998.

HSU, C. W.; CHANG, C. C.; LIN, C. J. ***A Pratical Guide to Support Vector Classification***.Taiwan. 2008
http://www.csie.ntu.edu.tw/~cjlin acesso em 11/02/2008

HUBNER, M. **Uma Proposta Para o Problema de Melhor Localização de Produtos na Área de Piching de um Armazém *(Slotting)***. Dissertação (Mestrado em Métodos Numéricos) – Universidade Federal do Paraná, PR, 2008.

IZMAILOV, A.; SOLODOV, M. **Otimização – Volume 1. Condições de Otimalidade, Elementos de Análise Convexa e Dualidade**. Rio de Janeiro: IMPA – Instituto Nacional de Matemática Pura e Aplicada, 2005.

JOHNSON, R. A., WICHERN, D. W., ***Applied Multivariate Statistical Analysis***. New Jersey: Editora Prentice Hall, Fourth Edition, 1998.

JOSHI, A.; CHANDRAN, S.; PHADKE, S.; JAYARAMAN, V.K.; KULKARNI, B.D. ***Arrhythmia Classification Using Local Holder Exponents and Support Vector Machine*.** India: Publish Springer Berlin, 2005.

LAI, K. K.; YU, L.; ZHOU, L.; WANG, S. ***Credit Risk Evalution with Least Square Support Vector Machine***. China: RSKT, 2006.

LEE, Y. J.; MANGASARIAN, O. L. ***RSVM: Reduced Support Vector Machine***. University of British Columbia. First SIAM, 2001.
http://www.cs.ubc.ca/local/reading/ acesso em 21/08/2008.

LI, Y.; GONG, S.; LIDDELL, H. ***Support Vector Regression and Classification Based Multi-view Face Detection and Recognition***. **In:** International Conference on Automatic Face and Gesture Recognition (IEEE), 4 . pg 300, United Kingdom, 2000.

MACHADO, A. L. F. **Pesquisa Operacional Aplicada à Análise de Portfólio**. Dissertação (Mestrado em Métodos Numéricos em Engenharia) – Universidade Federal do Paraná, PR, 2001.

MANGASARIAN, O. L., SETIONO, R., WOLBERG, W. H., ***Pattern Recognition Via Linear Programming: Theory and Application to Medical Diagnosis***. In: Large-Scale Numerical Optimization, Philadelphia: SIAM, pg 22-30 1990.

MANGASARIAN, O.L. ***Data Mining via Support Vector Machine***. In: Data Mining Institute Technical Report, 2001. IFIP Conference on System Modelling and Optimization, Germany, 2001.
http://citeseer.ist.psu.edu em 25/02/2008.

MURTY, K. G. ***Linear and Combinatorial Programming***. Florida: Editora Robert E. Krieger Publishing Company, 1985.

NGNYEN, H. N.; OHN, S. Y.; CHAE, S. H.; SONG, D. H.; LEE, I. ***Optimizing*** **Weighted Kernel Function for Support Vector Machine by Genetic Algorithm**. In: Lecture Notes in Computer Science . Springer Berlin / Heidelberg . Volume 4293, pg 583-592. 2006

ÖZGÜNDÜZ, E.; SENTÜRK, T.; KARSLIGIL, M. E. ***Off-Line Signature Verification and Recognition by Support Vector Machine***. Eusipco - 2005. Turkey. 2005.
http://www.eurasip.org/Proceedings/Eusipco/Eusipco2005 em 25/02/2008.

PAIXÃO, L. A. **Avaliação da qualidade do óleo isolante em transformadores com o emprego da função discriminante quadrática**. Dissertação (Mestrado em Métodos Numéricos) – Universidade Federal do Paraná, PR, 2006.

PETROUSTSOS, Evangelos***. Dominando o Visual Basic 6 – A Bíblia***. São Paulo: Makron Books, 1999.

PILLA JÚNIOR, Valfredo ; LOPES, H. S. . **Reconhecimento de Padrões em Sinais Eletrocardiográficos com Rede Reconhecimento de Padrões Eletroencefalográficos com Rede Neurofuzzy e Algoritmos Genéticos Algoritmos Genéticos.** In: Congresso Brasileiro de Redes Neurais, IV. São José dos Campos. Anais do IV Congresso Brasileiro de Redes Neurais, pg. 42-46, 1999.

PLATT, J. C. **Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines**. Microsoft Research. Technical Report MSR-TR-98-14. April 21,1998.
http://research.microsoft.com/users/jplatt/smoTR.pdf

RIBEIRO, A. A. **Convergência Global dos Métodos de Filtro para Programação não Linear**. Tese (Doutorado em Métodos Numéricos) – Universidade Federal do Paraná, PR, 2005.

RUAS, G. I. S. ; BRAGATTO, T. A. C. ; LAMAR, M. V. ; AOKI, A. R. ; ROCCO, S. M. **Previsão de Demanda de Energia Utilizando Redes Neurais Artificiais e *Support Vector Regression***. In: International Symposium on Communications, Control and Signal Processing, 3. Piscataway, NJ : v. 1. p. 1431-1435, 2008.

SCHÖLKOPF, B.; BURGES, C. J.C.; SMOLA, A. J. ***Advances in Kernel Methods: Support Vector Learning.*** Chapter 12 of John Platt. Mit Press Cambridge, MA, USA. 1999.

SHAWE-TAYLOR, J.; CRISTIANINI, N. ***Kernel Methods for Pattern Analysis.*** United Kingdom: Cambridge, 2004.

SILVA, F. F.; CANDEIAS, A. L. B. **Reconhecimento de Padrões para Mosaicagem e Geração de Imagens Digitais Maiores que o Formato A4.** In: Congresso Brasileiro de Cartografia, XXI. Belo Horizonte, v. 1, pg 1-6, 2003.

SOUZA, W. E. de. **Análise das Distorções Harmônicas de Tensão a Partir de Características dos Transformadores e de Dados de Consumo.** Dissertação (Mestrado em Métodos Numéricos para Engenharia) – Universidade Federal do Paraná, PR, 2008

STEINER, M. T. A. **Uma Metodologia para o Reconhecimento de Padrões Multivariados com Resposta Dicotômica**. Tese (Doutorado em Engenharia de Produção) – Universidade Federal de Santa Catarina, SC, 1995.

STOEAN, R. D. STOEAN, C. PREUSS, M. EL-DARZI, E. DUMITRESCU,D. **Evolutionary Support Vector Machines for Diabetes Mellitus Diagnosis.** In: Internacional Conference on Intelligent Systems, 3. London, pg 182-187, 2006.

TAN, P. N.; STEINBACH, M.; KUMAR, V.. **Introduction to Data Mining**. Inc. Boston, MA, USA: Addison-Wesley Longman Publishing Co., 2005.

Universidade Federal do Paraná. **Normas para apresentação de documentos científicos**: Teses, dissertações, monografias e trabalhos acadêmicos. Paraná, 2007.

Universidade Federal do Paraná. **Normas para apresentação de documentos científicos**: Citações e Notas de Rodapé. Paraná, 2007.

Universidade Federal do Paraná. **Normas para apresentação de documentos científicos**: Referências. Paraná, 2007.

VAPNIK, V. N., ***The Nature of Statistical Learning Theory***. New York: Springer, 1999.

ZAKI, M.; KARYPIS, G.; YANG, J.. **Data Mining in Bioinformatics (BIOKDD).** Algorithms for Molecular Biology, V 2, 2007.

APÊNDICE 1

TABELA 17 – DADOS DO PROBLEMA ICTERÍCIA

y_i	x^1	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^9	x^{10}	x^{11}	x^{12}	x^{13}	x^{14}
+1	46	1	41.8	21.20	20.60	234	178	646.25	92	14	3.3	0.80	9.0	36.8
+1	52	0	21.4	12.95	8.45	55	80	229.57	92	15	3.5	0.55	7.8	40.6
+1	73	0	26.2	13.60	12.60	90	97	116.38	104	14	2.7	0.80	12.6	32.3
+1	47	0	31.6	16.50	15.40	31	59	174.46	92	13	3.0	0.70	11.4	39.0
+1	66	0	40.0	20.90	19.10	45	108	366.74	66	11	3.6	0.80	9.2	30.3
+1	26	0	22.0	13.10	8.90	80	29	129.69	158	14	3.0	0.80	9.0	36.8
+1	66	1	25.6	14.00	11.60	125	129	300.74	100	13	2.7	0.90	8.1	40.5
+1	53	1	24.4	15.00	9.40	43	63	213.40	60	14	2.4	0.90	11.6	42.0
+1	34	0	19.8	11.60	8.20	24	47	70.51	92	13	3.6	0.80	9.0	36.8
+1	50	1	25.0	14.00	11.00	86	149	467.28	10	14	1.5	1.70	10.8	19.0
+1	69	1	11.9	7.55	4.35	176	92	240.68	104	13	3.4	1.00	7.5	40.0
+1	63	1	15.6	9.10	6.50	21	44	243.10	144	15	3.0	0.80	8.6	32.0
+1	43	1	13.7	7.70	6.00	25	63	286.00	79	13	3.2	0.70	7.0	34.0
+1	76	0	10.4	7.30	3.10	35	50	283.14	104	13	3.0	0.60	12.6	33.4
+1	66	1	19.8	10.70	9.10	48	68	283.14	104	14	3.0	0.80	9.0	36.8
+1	73	0	16.2	9.90	6.30	132	71	242.88	196	15	3.0	0.50	5.7	38.4
+1	46	1	8.6	5.60	3.00	28	56	283.14	104	15	3.0	1.00	9.0	36.8
+1	45	1	19.4	10.10	9.30	99	87	114.73	158	17	2.6	0.70	5.2	38.0
+1	60	1	18.8	10.10	9.70	33	92	264.77	158	17	2.3	0.70	7.5	39.5
+1	76	0	19.1	12.20	6.90	60	71	487.41	66	13	4.2	1.50	10.8	47.9
+1	33	0	3.8	2.40	1.40	35	61	234.63	60	13	3.0	0.70	5.3	34.2
+1	46	1	3.0	2.20	0.80	390	400	725.45	132	15	3.0	1.00	8.0	44.5
+1	55	1	13.4	6.85	6.55	45	97	108.35	123	14	2.7	0.70	11.1	36.7
+1	38	1	34.2	18.40	15.80	265	265	381.48	92	13	3.0	0.50	7.9	39.1
+1	68	1	11.8	7.60	4.20	188	192	132.66	158	14	3.0	0.80	8.3	40.8
+1	73	1	26.9	14.00	12.90	86	102	328.46	100	15	2.7	1.00	7.7	28.6
+1	57	0	17.4	10.00	7.40	37	42	229.46	197	13	3.4	0.95	12.0	39.8
+1	60	1	16.8	9.70	7.10	104	92	165.00	82	19	3.0	0.60	10.5	37.4
+1	59	0	19.4	13.59	5.79	132	113	220.11	88	14	3.0	0.90	7.0	38.2
+1	73	0	30.0	18.40	11.60	75	107	414.70	60	13	3.3	0.60	11.3	34.0
+1	76	1	29.4	16.80	12.60	78	91	368.83	106	13	3.2	0.80	9.0	36.8
+1	75	1	14.5	10.28	4.17	210	370	1050.00	88	15	3.5	1.50	8.7	45.0
+1	53	1	24.2	19.37	4.86	78	92	1525.00	79	14	3.3	0.70	9.8	43.0
+1	55	0	17.7	13.38	4.30	19	47	1027.00	123	14	2.4	0.70	9.3	32.7
+1	28	0	32.2	21.35	10.88	24	59	744.00	115	19	2.96	1.20	28.0	22.4
-1	74	1	34.8	18.80	16.00	50	56	145.64	172	12	3.0	1.00	7.7	45.5
-1	47	0	24.0	14.00	10.00	205	225	305.69	186	13	3.8	0.95	10.0	40.3
-1	82	0	24.4	13.45	10.95	16	68	248.60	82	13	3.0	0.80	7.5	39.8
-1	56	0	34.8	19.10	15.70	43	29	46.75	79	16	3.0	1.90	22.0	43.0
-1	80	1	10.0	5.50	4.50	12	32	74.36	88	12	1.9	0.80	8.8	38.0
-1	46	0	11.4	7.00	4.40	78	40	150.04	79	14	3.7	0.80	11.0	37.0
-1	55	0	8.4	5.50	2.90	19	53	106.70	100	14	2.3	1.00	6.3	48.7
-1	17	0	7.3	4.55	2.75	40	41	151.47	164	13	3.0	0.50	6.2	34.8
-1	61	0	15.4	9.90	5.50	156	55	124.85	92	13	3.0	0.50	9.0	38.0
-1	26	0	3.0	2.30	0.70	147	216	150.04	109	13	3.0	0.80	8.8	44.0
-1	61	0	12.5	7.00	5.50	19	45	109.89	88	13	2.9	0.90	9.2	38.2
-1	58	1	13.3	8.10	5.20	104	83	262.13	142	13	4.1	0.90	11.9	42.9

Continua

Continuação

y_i	x^1	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^9	x^{10}	x^{11}	x^{12}	x^{13}	x^{14}
-1	23	1	8.0	1.00	7.00	6	25	27.50	66	14	5.5	0.60	6.4	31.0
-1	48	1	4.6	2.60	2.00	53	47	91.96	880	14	3.8	1.10	10.5	45.0
-1	38	0	4.6	3.20	1.40	55	66	298.10	79	13	3.0	0.80	6.1	45.8
-1	66	0	6.9	5.10	1.80	250	100	59.40	65	13	2.0	1.30	10.0	37.4
-1	49	1	2.6	1.20	1.40	26	39	124.63	225	13	1.7	0.80	4.7	41.9
-1	47	0	4.8	3.20	1.60	22	19	35.64	82	15	1.7	1.40	7.5	11.6
-1	28	0	8.2	4.90	3.30	93	216	93.83	540	14	2.2	1.50	11.8	37.9
-1	34	0	14.6	9.10	5.50	290	200	215.16	88	14	3.6	0.70	6.0	36.2
-1	56	0	3.2	1.60	1.60	33	50	189.09	79	15	3.0	0.90	19.4	49.0
-1	25	0	9.2	6.20	3.00	670	1220	103.62	66	14	3.7	0.70	8.3	46.0
-1	42	0	2.4	1.30	1.10	215	145	150.04	100	14	3.0	0.65	3.7	39.8
-1	45	1	2.4	1.40	1.00	22	36	88.99	66	13	2.0	2.50	5.4	35.0
-1	26	0	2.8	2.40	0.40	99	74	110.00	186	14	1.7	1.80	16.0	31.3
-1	34	0	8.2	5.95	2.25	64	71	297.88	66	23	3.4	0.95	7.9	44.0
-1	38	1	2.8	1.20	1.60	41	40	77.55	88	12	3.6	0.90	6.0	45.0
-1	60	0	4.8	3.40	1.40	21	44	132.55	88	14	3.0	0.80	6.4	42.3
-1	55	0	8.0	5.00	3.00	17	47	145.64	92	13	2.7	1.00	6.0	36.5
-1	73	0	11.9	7.80	4.10	82	92	43.56	186	15	3.0	1.20	6.0	47.0
-1	65	0	6.8	3.30	3.50	35	59	105.16	305	12	3.3	3.15	6.8	36.5
-1	63	1	4.4	2.80	1.60	21	55	182.93	186	12	1.7	1.20	9.6	39.1
-1	56	1	3.6	2.10	1.50	6	19	80.85	186	14	2.3	0.80	8.6	31.0
-1	27	1	6.4	4.20	2.20	30	63	66.33	1648	14	3.0	0.60	10.3	34.2
-1	42	1	10.2	4.55	5.65	53	57	458.37	82	13	2.6	0.90	15.7	36.2
-1	65	0	2.0	0.95	1.05	35	24	318.56	186	12	3.0	1.70	8.6	32.4
-1	73	0	6.3	4.60	1.70	12	36	150.04	186	14	3.0	1.20	20.5	36.6
-1	25	0	6.8	4.65	2.15	78	61	80.85	52	12	3.5	0.70	6.0	40.5
-1	24	0	11.4	6.10	5.30	16	27	98.67	210	13	3.9	0.65	5.1	43.1
-1	78	1	19.4	10.70	8.70	53	50	337.70	60	13	3.5	0.80	7.5	45.0
-1	77	0	3.6	2.20	1.40	16	20	110.00	100	13	3.0	2.00	10.0	34.0
-1	67	0	6.8	5.35	1.45	7	27	120.01	255	13	3.0	0.80	13.4	35.3
-1	58	1	34.8	18.80	16.00	216	113	230.56	82	13	4.5	0.50	7.0	45.5
-1	24	0	6.0	2.00	4.00	100	108	148.83	60	13	3.3	0.60	12.0	33.0
-1	49	1	13.9	7.99	5.93	140	265	206.91	60	14	3.8	1.20	6.4	42.7
-1	38	1	16.2	9.55	6.65	42	53	189.20	482	16	3.0	0.40	5.1	37.6
-1	30	0	6.3	5.10	1.20	190	89	158.62	79	14	3.4	1.10	10.5	48.0
-1	34	0	2.0	1.20	0.80	62	45	333.30	88	14	3.6	0.95	9.6	39.1
-1	69	1	4.6	2.60	2.00	68	66	195.91	60	14	3.0	0.90	7.8	46.4
-1	43	0	2.0	0.75	1.25	28	59	158.51	82	14	3.0	0.70	5.1	37.7
-1	25	0	5.8	2.90	2.90	74	78	187.77	352	17	2.6	0.60	27.1	19.0
-1	72	1	9.5	5.20	4.30	106	133	158.62	600	14	3.0	1.20	16.7	41.0
-1	24	0	3.9	3.21	0.75	182	115	158.62	60	14	3.4	0.96	6.7	39.1
-1	48	0	2.4	0.60	1.80	15	60	29.15	92	14	4.2	0.80	9.6	39.1
-1	83	1	3.2	1.60	1.60	28	10	58.30	123	14	2.8	1.25	13.3	39.1
-1	47	0	4.3	3.10	1.20	69	98	92.07	176	16	3.1	0.70	9.6	39.1
-1	34	0	6.4	4.10	2.30	17	47	388.19	186	14	3.4	0.90	10.0	44.0
-1	72	0	0.8	0.30	0.50	6	18	71.06	186	13	2.5	0.90	9.6	39.1
-1	60	1	18.8	10.10	8.70	33	92	264.77	186	17	2.3	0.70	7.5	39.5
-1	63	1	8.2	4.75	3.45	10	55	182.93	186	13	1.7	1.20	9.0	39.1
-1	31	0	2.8	1.98	0.82	290	110	292.00	204	14	3.02	0.83	10.1	37.9
-1	34	0	2.4	0.40	2.00	40	33	86.00	60	14	3.20	0.83	10.1	37.9
-1	53	0	14.9	9.20	5.75	210	83	199.78	189	16	3.40	0.70	7.8	45.5

Continua

Conclusão

y_i	x^1	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^9	x^{10}	x^{11}	x^{12}	x^{13}	x^{14}
-1	37	0	3.4	0.97	2.40	132	77	1008.00	814	17	3.10	1.00	5.8	41.0
-1	31	0	5.3	3.87	1.41	66	55	207.00	88	14	3.70	0.80	6.3	40.0
-1	36	0	7.1	4.45	2.65	176	104	338.00	60	14	2.90	0.80	13.8	40.0
-1	55	0	2.6	0.97	1.63	290	50	199.78	56	13	2.90	0.70	4.7	39.0
-1	30	0	31.1	22.00	9.10	50	71	508.00	417	15	2.60	1.70	9.7	18.7
-1	61	0	1.8	0.84	1.01	26	29	406.00	273	15	3.30	0.50	14.5	40.5
-1	54	0	2.3	1.14	1.21	216	94	491.00	66	13	3.20	0.80	7.2	41.1
-1	81	1	1.6	0.65	0.95	19	21	449.00	142	15	3.30	1.20	8.1	38.0
-1	50	0	7.8	6.10	1.70	102	95	592.00	79	13	3.00	0.60	10.5	46.5
-1	51	0	9.5	6.19	3.31	125	66	300.00	1362	13	2.80	0.40	7.2	45.0
-1	39	0	36.7	28.50	8.16	230	290	2972.00	382	14	2.50	0.30	13.0	35.6
-1	48	1	1.9	0.85	1.05	80	65	158.62	60	15	3.30	1.00	9.0	39.0
-1	34	0	6.5	4.60	1.90	180	230	158.62	79	14	2.90	0.80	11.2	45.0
-1	27	0	29.2	19.38	9.86	300	320	158.62	18	14	3.10	0.50	8.3	43.0
-1	63	1	2.8	1.60	1.20	16	16	214.00	189	15	3.10	0.90	10.3	40.0
-1	45	0	12.2	10.20	2.00	10	20	158.62	76	13	2.20	0.70	8.9	42.0
-1	50	0	8.1	5.69	2.41	16	29	158.62	189	13	2.40	0.80	8.4	38.0
-1	39	1	7.9	5.09	2.86	20	27	158.62	219	12	3.00	3.30	10.3	33.6
-1	66	1	4.0	2.00	2.00	20	53	285.00	76	18	2.80	1.20	10.7	44.0
-1	29	1	1.0	0.50	0.50	104	57	370.00	189	15	3.50	0.80	8.3	44.2

FONTE: Steiner (1995)

As características e mais informações podem ser encontradas em Steiner (1995).

APÊNDICE 2

PROBLEMA ICTERÍCIA SEM TRATAMENTO DOS DADOS

O problema da Icterícia foi treinado de duas formas: dados sem nenhum tratamento e dados normalizados, devido a diferença de grandezas entre as características consideradas na coleta.

No treinamento dos dados do problema da Icterícia, com as unidades próprias de cada característica, não se obteve bons resultados, como pode ser visto nas tabelas a seguir.

Uma observação importante é que utilizando o kernel polinomial não homogêneo de grau superior a dois não ocorreu nenhuma iteração, pois os parâmetros não estão adequados aos dados, com isso não violam as condições de KKT. Na tabela abaixo, não ocorreram iterações no kernel sigmoidal também.

TABELA 18 – TREINAMENTO DOS DADOS ICTERÍCIA

Nº Iterações	Nº VS	Nº VS bound	Erros %	C	Kernel	p	Constante k	Kapa κ	Sigma σ	Tempo (s)
35	0	70	0,00%	0,1	Sigmoidal	0	0	0.001	0	0,0630625
35	0	70	0,00%	1	Sigmoidal	0	0	0.01	0	6,35E-02
1422	83	35	0,00%	1	Gaussiano	0	0	0	0.001	1,01675
1530	83	35	0,00%	1	Gaussiano	0	0	0	0.01	1,172375
1363	83	35	0,00%	1	Gaussiano	0	0	0	0.1	0,96875
1309	83	35	0,00%	1	Gaussiano	0	0	0	1	0,95225
1534	83	35	0,00%	1	Gaussiano	0	0	0	10	1,497125
383	118	0	0,00%	10	Gaussiano	0	0	0	0.001	0,5005
383	118	0	0,00%	10	Gaussiano	0	0	0	0.01	0,49975
383	118	0	0,00%	10	Gaussiano	0	0	0	0.1	0,499375
383	118	0	0,00%	10	Gaussiano	0	0	0	1	0,498625
383	118	0	0,00%	10	Gaussiano	0	0	0	10	0,48325
35	0	70	0,00%	100	Sigmoidal	0	0	0.01	0	7,66E-02
383	118	0	0,00%	100	Gaussiano	0	0	0	0.001	0,497875
383	118	0	0,00%	100	Gaussiano	0	0	0	0.01	0,4975
383	118	0	0,00%	100	Gaussiano	0	0	0	0.1	0,5016562
383	118	0	0,00%	100	Gaussiano	0	0	0	1	0,5012813
383	118	0	0,00%	100	Gaussiano	0	0	0	10	0,5005313
35	0	70	0,00%	1000	Sigmoidal	0	0	0.01	0	0,0643125
383	118	0	0,00%	1000	Gaussiano	0	0	0	0.001	0,5018438
383	118	0	0,00%	1000	Gaussiano	0	0	0	0.01	0,5010937
383	118	0	0,00%	1000	Gaussiano	0	0	0	0.1	0,5167187
383	118	0	0,00%	1000	Gaussiano	0	0	0	1	0,5313438

Continua

Nº Iterações	Nº VS	Nº VS bound	Erros %	C	Kernel	p	Constante k	Kapa κ	Sigma σ	Tempo (s)
383	118	0	0,00%	1000	Gaussiano	0	0	0	10	0,5003437
35	0	70	0,00%	0,001	Sigmoidal	0	0	0.001	0	0,093375
35	0	70	0,00%	0,01	Sigmoidal	0	0	0.001	0	7,70E-02
35	0	70	0,85%	0,1	Sigmoidal	0	-0.5	0.001	0	0,0634375
35	0	70	0,85%	0,1	Sigmoidal	0	-0.75	0.001	0	0,0634375
35	0	70	0,85%	0,1	Sigmoidal	0	-1	0.001	0	0,0638125
35	0	70	0,85%	1	Sigmoidal	0	0	0.001	0	0,0600625
35	0	70	0,85%	1	Sigmoidal	0	-0.25	0.001	0	6,43E-02
35	0	70	1,70%	1	Sigmoidal	0	-0.5	0.001	0	6,43E-02
35	0	70	1,70%	1	Sigmoidal	0	-0.75	0.001	0	6,41E-02
35	0	70	1,70%	1	Sigmoidal	0	-1	0.001	0	6,41E-02
35	0	70	1,70%	10	Sigmoidal	0	-0.25	0.001	0	7,81E-02
35	0	70	1,70%	10	Sigmoidal	0	-0.5	0.001	0	7,88E-02
35	0	70	1,70%	100	Sigmoidal	0	0	0.001	0	7,68E-02
35	0	70	1,70%	100	Sigmoidal	0	-0.25	0.001	0	7,66E-02
35	0	70	1,70%	100	Sigmoidal	0	-0.5	0.001	0	7,63E-02
35	0	70	1,70%	1000	Sigmoidal	0	0	0.001	0	6,02E-02
35	0	70	1,70%	1000	Sigmoidal	0	-0.25	0.001	0	0,0644375
35	0	70	1,70%	1000	Sigmoidal	0	-0.5	0.001	0	0,0638125
35	0	70	2,54%	10	Sigmoidal	0	-0.75	0.001	0	7,82E-02
35	0	70	2,54%	100	Sigmoidal	0	-0.75	0.001	0	7,61E-02
35	0	70	2,54%	1000	Sigmoidal	0	-0.75	0.001	0	0,0645625
35	0	70	3,39%	10	Sigmoidal	0	-1	0.001	0	7,80E-02
35	0	70	3,39%	100	Sigmoidal	0	-1	0.001	0	0,07975
35	0	70	3,39%	1000	Sigmoidal	0	-1	0.001	0	0,0639375
23600	27	25	16,95%	0,01	Polinomial	1	0.75	0	0	18,44119

FONTE: A autora (2008)

APÊNDICE 3

PROBLEMA ICTERÍCIA COM DADOS NORMALIZADOS

Em virtude dos resultados fracos, normalizaram-se os dados, já que os mesmo possuíam unidades muito diferenciadas, como por exemplo, a característica sexo possuía entrada dicotômica 0 ou 1, idade possuía resposta inteira, mas com unidade de grandeza maior, o dados provenientes de exames por sua vez, possuíam respostas contínuas mas em unidades variadas.

Ao contrário dos dados não tratados, os resultados foram bons. Em alguns casos, os parâmetros exigiram mais iteração do algoritmo, chegando ao máximo que foi fixado em 23600 (118x200).

TABELA 19 – TREINAMENTO DOS DADOS ICTERÍCIA

Nº Iterações	Nº VS	Nº VS bound	Erros %	C	Kernel	p	Constante k	Kapa κ	Sigma σ	Tempo (s)
35	0	70	0,00%	0,001	Sigmoidal	0	0	10	0	6,25E-02
35	0	70	0,00%	0,001	Sigmoidal	0	0	100	0	6,31E-02
35	0	70	0,00%	0,001	Sigmoidal	0	0	1000	0	6,33E-02
35	0	70	0,00%	0,01	Sigmoidal	0	0	10	0	0,0634375
35	0	70	0,00%	0,01	Sigmoidal	0	0	100	0	0,0640625
35	0	70	0,00%	0,01	Sigmoidal	0	0	1000	0	0,0790625
35	0	70	0,00%	0,1	Sigmoidal	0	0	10	0	0,0610625
35	0	70	0,00%	0,1	Sigmoidal	0	0	100	0	0,0600625
35	0	70	0,00%	0,1	Sigmoidal	0	0	1000	0	6,30E-02
23600	37	1	0,00%	1	Polinomial	6	1	0	0	16,79575
35	0	70	0,00%	1	Sigmoidal	0	0	10	0	6,29E-02
35	0	70	0,00%	1	Sigmoidal	0	0	100	0	0,06175
35	0	70	0,00%	1	Sigmoidal	0	0	1000	0	6,22E-02
572	29	65	0,00%	1	Gaussiano	0	0	0	0.001	0,4376875
19820	38	1	0,00%	10	Polinomial	5	0.25	0	0	15,01891
23600	37	0	0,00%	10	Polinomial	5	0.5	0	0	17,03925
20154	37	0	0,00%	10	Polinomial	5	1	0	0	15,89575
23600	36	0	0,00%	10	Polinomial	6	0.25	0	0	16,74328
23497	37	0	0,00%	10	Polinomial	6	0.5	0	0	17,66834
23600	33	0	0,00%	10	Polinomial	7	0.25	0	0	16,69644
23507	34	0	0,00%	10	Polinomial	7	0.5	0	0	17,129
18015	36	0	0,00%	10	Polinomial	7	0.75	0	0	13,13097

Continua

Nº Iterações	Nº VS	Nº VS bound	Erros %	C	Kernel	p	Constante k	Kapa κ	Sigma σ	Tempo (s)
35	0	70	0,00%	10	Sigmoidal	0	0	10	0	6,43E-02
35	0	70	0,00%	10	Sigmoidal	0	0	100	0	0,0625
35	0	70	0,00%	10	Sigmoidal	0	0	1000	0	0,064875
521	118	0	0,00%	10	Gaussiano	0	0	0	0.001	0,4387187
23600	39	0	0,00%	100	Polinomial	4	0.5	0	0	17,33281
23344	39	0	0,00%	100	Polinomial	4	0.75	0	0	23,35603
23600	39	0	0,00%	100	Polinomial	4	1	0	0	17,26881
23253	38	0	0,00%	100	Polinomial	5	0.5	0	0	18,35322
23478	38	0	0,00%	100	Polinomial	5	0.75	0	0	17,68262
16039	35	0	0,00%	100	Polinomial	6	0	0	0	12,44994
22304	36	0	0,00%	100	Polinomial	6	0.75	0	0	16,41966
21891	33	0	0,00%	100	Polinomial	7	0.25	0	0	15,98009
18466	36	0	0,00%	100	Polinomial	7	0.75	0	0	13,37253
35	0	70	0,00%	100	Sigmoidal	0	0	10	0	0,061625
35	0	70	0,00%	100	Sigmoidal	0	0	100	0	0,06275
35	0	70	0,00%	100	Sigmoidal	0	0	1000	0	0,0635
521	118	0	0,00%	100	Gaussiano	0	0	0	0.001	0,423625
643	118	0	0,00%	100	Gaussiano	0	0	0	0.01	0,471375
23600	38	0	0,00%	1000	Polinomial	3	0.25	0	0	16,56325
23600	40	0	0,00%	1000	Polinomial	3	1	0	0	16,6015
23600	38	0	0,00%	1000	Polinomial	4	0	0	0	16,71119
23600	39	0	0,00%	1000	Polinomial	4	0.5	0	0	16,85131
23600	39	0	0,00%	1000	Polinomial	4	0.75	0	0	17,337
21246	38	0	0,00%	1000	Polinomial	5	0	0	0	16,39862
22101	38	0	0,00%	1000	Polinomial	5	0.25	0	0	16,83144
20216	38	0	0,00%	1000	Polinomial	5	0.5	0	0	15,33966
17343	38	0	0,00%	1000	Polinomial	5	0.75	0	0	13,40706
23600	34	0	0,00%	1000	Polinomial	7	0.5	0	0	16,72972
35	0	70	0,00%	1000	Sigmoidal	0	0	10	0	6,37E-02
35	0	70	0,00%	1000	Sigmoidal	0	0	100	0	6,28E-02
35	0	70	0,00%	1000	Sigmoidal	0	0	1000	0	6,29E-02
521	118	0	0,00%	1000	Gaussiano	0	0	0	0.001	0,4224063
657	118	0	0,00%	1000	Gaussiano	0	0	0	0.01	0,4695312
2558	84	34	0,85%	1	Gaussiano	0	0	0	0.01	1,931813
23063	40	1	0,85%	10	Polinomial	5	0	0	0	18,30963
1281	116	2	0,85%	10	Gaussiano	0	0	0	0.01	0,9888437
23600	39	0	0,85%	100	Polinomial	3	0.5	0	0	17,11256
23600	37	0	0,85%	100	Polinomial	3	1	0	0	17,11087
23600	36	0	0,85%	100	Polinomial	4	0	0	0	16,53737
23600	33	0	0,85%	100	Polinomial	6	0.5	0	0	16,90375
23600	34	0	0,85%	1000	Polinomial	4	0.25	0	0	16,5385
23600	36	0	0,85%	1000	Polinomial	4	1	0	0	16,60231
23600	33	9	1,70%	0,1	Polinomial	6	0.25	0	0	16,82725
23600	32	7	1,70%	0,1	Polinomial	6	0.5	0	0	16,84
23600	38	4	1,70%	0,1	Polinomial	6	1	0	0	16,79406
23600	38	1	1,70%	0,1	Polinomial	7	0.5	0	0	16,65275
23600	38	1	1,70%	1	Polinomial	6	0.25	0	0	16,81341

Continua

Nº Iterações	Nº VS	Nº VS bound	Erros %	C	Kernel	p	Constante k	Kapa κ	Sigma σ	Tempo (s)
23600	33	0	1,70%	1000	Polinomial	2	0.5	0	0	16,61128
23600	38	0	1,70%	1000	Polinomial	3	0.5	0	0	16,595
23600	31	0	1,70%	1000	Polinomial	7	0.25	0	0	16,69678
23600	28	9	2,54%	1	Polinomial	4	1	0	0	17,83753
23600	35	1	2,54%	1	Polinomial	5	1	0	0	17,23753
23600	33	9	2,54%	10	Polinomial	3	0.5	0	0	16,90756
23600	32	10	2,54%	10	Polinomial	3	0.75	0	0	17,37894
23600	33	9	2,54%	10	Polinomial	3	1	0	0	16,98991
23600	36	2	2,54%	10	Polinomial	4	0.25	0	0	17,13291
23600	33	0	2,54%	10	Polinomial	7	0	0	0	16,83713
23600	38	0	2,54%	100	Polinomial	5	0	0	0	17,39953
23600	33	0	2,54%	1000	Polinomial	2	0	0	0	16,61041
23600	34	0	2,54%	1000	Polinomial	2	0.75	0	0	16,58078
23600	34	0	2,54%	1000	Polinomial	6	0.5	0	0	16,57384
23600	37	0	2,54%	1000	Polinomial	6	0.75	0	0	16,57187
23600	36	1	3,39%	0,1	Polinomial	7	1	0	0	16,79313
23600	32	9	3,39%	1	Polinomial	4	0.75	0	0	17,01528
23600	29	6	3,39%	1	Polinomial	5	0	0	0	16,96856
23600	34	5	3,39%	1	Polinomial	5	0.25	0	0	17,45869
23600	37	1	3,39%	1	Polinomial	6	0.5	0	0	16,76678
23600	33	1	3,39%	1	Polinomial	6	0.75	0	0	16,77913
23600	30	10	3,39%	10	Polinomial	3	0	0	0	16,79781
23600	29	9	3,39%	10	Polinomial	3	0.25	0	0	16,93919
23600	36	2	3,39%	10	Polinomial	4	0.5	0	0	16,96134
23600	34	2	3,39%	10	Polinomial	4	0.75	0	0	17,52703
23600	33	0	3,39%	10	Polinomial	7	1	0	0	16,70394
23600	34	8	3,39%	100	Polinomial	2	1	0	0	16,95538
23600	35	0	3,39%	100	Polinomial	4	0.25	0	0	16,76819
23600	33	0	3,39%	100	Polinomial	6	1	0	0	16,62394
23600	35	0	3,39%	1000	Polinomial	2	1	0	0	16,56162
23600	36	0	3,39%	1000	Polinomial	6	0	0	0	16,60122
23600	34	0	3,39%	1000	Polinomial	6	0.25	0	0	16,60353
14175	29	4	3,39%	1000	Gaussiano	0	0	0	0.1	10,20278
11595	31	15	4,24%	0,1	Polinomial	5	0.5	0	0	8,396719
23600	27	13	4,24%	0,1	Polinomial	6	0	0	0	37,55669
19949	31	15	4,24%	0,1	Polinomial	5	0.5	0	0	14,39256
23600	26	11	4,24%	1	Polinomial	4	0.5	0	0	16,87631
23600	34	0	4,24%	10	Polinomial	6	0.75	0	0	16,66463
23600	33	8	4,24%	100	Polinomial	2	0.5	0	0	16,62806
23600	31	8	4,24%	100	Polinomial	2	0.75	0	0	16,62803
23600	37	0	4,24%	1000	Polinomial	3	0	0	0	16,60938
23600	33	0	4,24%	1000	Polinomial	3	0.75	0	0	16,6575
23600	33	0	4,24%	1000	Polinomial	6	1	0	0	16,58881
21620	27	15	5,09%	0,1	Polinomial	5	0.75	0	0	15,64019
23600	32	6	5,09%	0,1	Polinomial	7	0	0	0	16,96697
23267	36	1	5,09%	0,1	Polinomial	7	0.25	0	0	16,55809
20971	27	15	5,09%	1	Polinomial	4	0	0	0	15,47466

Continua

Nº Iterações	Nº VS	Nº VS bound	Erros %	C	Kernel	p	Constante k	Kapa κ	Sigma σ	Tempo (s)
23600	31	13	5,09%	1	Polinomial	4	0.25	0	0	17,36297
23600	35	2	5,09%	1	Polinomial	5	0.75	0	0	17,45959
23600	34	1	5,09%	1	Polinomial	7	0.25	0	0	16,71688
23600	34	1	5,09%	10	Polinomial	4	0	0	0	16,9785
17074	49	9	5,09%	10	Gaussiano	0	0	0	0.1	12,58256
23600	32	8	5,09%	100	Polinomial	2	0	0	0	16,61516
23600	32	8	5,09%	100	Polinomial	2	0.25	0	0	16,59975
23600	35	0	5,09%	100	Polinomial	3	0	0	0	17,26659
5039	31	6	5,09%	100	Gaussiano	0	0	0	0.1	3,751719
7363	27	11	5,93%	0,01	Polinomial	7	0.5	0	0	5,079219
19453	24	27	5,93%	0,1	Polinomial	4	0.75	0	0	14,18225
23600	25	14	5,93%	0,1	Polinomial	6	0	0	0	16,79434
22208	25	25	5,93%	1	Polinomial	3	0.5	0	0	16,11469
23600	34	3	5,93%	1	Polinomial	5	0.5	0	0	17,363
23600	35	0	5,93%	1	Polinomial	7	0.75	0	0	24,51894
23600	23	23	5,93%	10	Polinomial	2	1	0	0	16,75094
23600	35	0	5,93%	10	Polinomial	6	1	0	0	16,72681
23600	35	0	5,93%	100	Polinomial	3	0.25	0	0	16,77022
23600	37	0	5,93%	100	Polinomial	5	1	0	0	16,80728
3664	23	20	6,78%	0,01	Polinomial	6	0.5	0	0	2,56275
3976	25	13	6,78%	0,01	Polinomial	6	1	0	0	2,76775
5083	30	8	6,78%	0,01	Polinomial	7	0.75	0	0	3,533969
20638	27	19	6,78%	0,1	Polinomial	5	0.25	0	0	15,02137
18321	20	27	6,78%	0,1	Polinomial	4	1	0	0	13,03159
18448	22	18	6,78%	0,1	Polinomial	5	0.25	0	0	13,34506
23600	25	13	6,78%	0,1	Polinomial	5	1	0	0	17,04344
17031	32	1	6,78%	0,1	Polinomial	7	0.75	0	0	12,33028
20693	23	25	6,78%	1	Polinomial	3	0.75	0	0	15,07166
23600	15	26	6,78%	1	Polinomial	3	1	0	0	17,02422
7340	39	41	6,78%	1	Gaussiano	0	0	0	0.1	5,305344
23600	20	25	6,78%	10	Polinomial	2	0.5	0	0	16,88806
21610	24	24	6,78%	10	Polinomial	2	0.75	0	0	17,65731
23600	29	0	6,78%	100	Polinomial	7	0.5	0	0	16,5745
23600	35	0	6,78%	1000	Polinomial	7	1	0	0	16,54206
1975	12	37	7,63%	0,01	Polinomial	5	0.5	0	0	1,406906
2947	22	26	7,63%	0,01	Polinomial	6	0	0	0	2,31175
2037	20	23	7,63%	0,01	Polinomial	6	0.25	0	0	1,406125
3647	25	18	7,63%	0,01	Polinomial	6	0.75	0	0	2,548125
3124	24	13	7,63%	0,01	Polinomial	7	0.25	0	0	2,171219
4676	28	4	7,63%	0,01	Polinomial	7	1	0	0	3,265063
23600	20	15	7,63%	0,1	Polinomial	5	0.75	0	0	26,64506
23600	21	14	7,63%	0,1	Polinomial	5	1	0	0	27,55841
22081	19	30	7,63%	0,1	Polinomial	4	0.5	0	0	15,81616
18787	20	30	7,63%	0,1	Polinomial	4	0.75	0	0	13,50263
23600	30	4	7,63%	0,1	Polinomial	6	0.75	0	0	17,59003
23600	19	27	7,63%	1	Polinomial	3	0.25	0	0	16,72537
23600	31	0	7,63%	1	Polinomial	7	1	0	0	28,39728

Continua

Nº Iterações	Nº VS	Nº VS bound	Erros %	C	Kernel	p	Constante k	Kapa κ	Sigma σ	Tempo (s)
18130	19	27	7,63%	10	Polinomial	2	0	0	0	13,16894
23600	31	0	7,63%	10	Polinomial	5	0.75	0	0	16,61706
23600	38	0	7,63%	100	Polinomial	3	0.75	0	0	16,75353
23600	35	0	7,63%	100	Polinomial	7	1	0	0	16,59231
20514	20	34	8,48%	0,1	Polinomial	4	0.25	0	0	14,60137
20683	15	35	8,48%	0,1	Polinomial	4	0.5	0	0	14,56969
17580	25	23	8,48%	0,1	Polinomial	4	1	0	0	12,86544
21097	21	22	8,48%	0,1	Polinomial	5	0	0	0	15,26713
22335	21	34	8,48%	0,1	Polinomial	4	0.25	0	0	16,80006
13835	20	20	8,48%	0,1	Polinomial	5	0	0	0	9,958219
23600	15	32	8,48%	1	Polinomial	3	0	0	0	16,77394
23600	29	1	8,48%	10	Polinomial	4	1	0	0	16,63156
23600	34	0	8,48%	100	Polinomial	5	0.25	0	0	16,87334
23600	29	0	8,48%	1000	Polinomial	2	0.25	0	0	16,53228
23600	32	0	8,48%	1000	Polinomial	5	1	0	0	16,63569
23600	15	30	8,48%	1000	Sigmoidal	0	-1	0.1	0	16,68697
2919	13	37	9,32%	0,01	Polinomial	5	0.75	0	0	2,064531
23600	32	1	9,32%	1	Polinomial	6	0	0	0	18,18312
23600	19	22	9,32%	10	Polinomial	2	0.25	0	0	16,84484
23600	32	0	9,32%	100	Polinomial	6	0.25	0	0	18,48097
23600	35	0	9,32%	1000	Polinomial	7	0	0	0	16,55522
958	12	35	10,17%	0,001	Polinomial	6	1	0	0	0,67125

FONTE: A autora (2008)

APÊNDICE 4

DADOS *ADULT*

Os dados *Adult* A1A, contendo 1605 pontos, foi treinado com várias funções *kernel* e diferenciados parâmetros. Os parâmetros que apresentarem erros inferiores a 10% durante o treinamento encontram-se na tabela a seguir:

TABELA 20 – TREINAMENTO DOS DADOS *ADULT* A1A

Nº Iterações	Nº VS	Nº VS *bound*	Erros %	C	Kernel	p	Constante k	Kapa κ	Sigma σ
417	0	790	0,13%	10	Sigmoidal	-	-0.75	10	-
430	0	790	0,13%	1000	Sigmoidal	-	-	1000	-
428	0	790	0,13%	1000	Sigmoidal	-	-0.75	1000	-
6159	336	730	0,87%	1	Gaussiano	-	-	-	0.001
5911	310	729	0,87%	1	Gaussiano	-	-	-	0.01
6053	328	731	0,87%	1	Gaussiano	-	-	-	0.1
23184	1360	28	0,87%	10	Gaussiano	-	-	-	0.001
18881	1369	28	0,87%	10	Gaussiano	-	-	-	0.01
19285	1350	28	0,87%	10	Gaussiano	-	-	-	0.1
24199	1457	28	0,87%	10	Gaussiano	-	-	-	1
84275	1112	28	0,94%	100	Gaussiano	-	-	-	0.001
82564	1072	28	0,94%	100	Gaussiano	-	-	-	0.1
135494	1088	28	0,94%	100	Gaussiano	-	-	-	1
321000	823	24	1,06%	1000	Gaussiano	-	-	-	0.001
52737	955	28	1,12%	100	Gaussiano	-	-	-	0.01
7443	655	2	1,25%	0,1	Polinomial	5	0.5	-	-
2283	741	1	1,31%	0,001	Polinomial	6	0.5	-	-
2522	747	3	1,31%	0,01	Polinomial	6	-	-	-
321000	783	28	1,31%	1000	Polinomial	6	1	-	-
2493	759	0	1,37%	0,0001	Polinomial	6	0	-	-
7533	662	2	1,37%	1	Polinomial	5	0.5	-	-
8763	556	679	1,37%	1	Gaussiano	-	-	-	1
2424	760	3	1,43%	1	Polinomial	6	0.5	-	-
2136	776	1	1,43%	1	Polinomial	6	1	-	-
2515	742	3	1,50%	0,1	Polinomial	6	0.5	-	-
6546	672	0	1,50%	10	Polinomial	5	0.25	-	-
6362	670	0	1,50%	1000	Polinomial	5	0.75	-	-
8144	661	2	1,56%	0,0001	Polinomial	5	0.25	-	-
2432	752	4	1,56%	1	Polinomial	6	0.25	-	-
13534	629	4	1,56%	10	Polinomial	5	1	-	-
7832	661	4	1,62%	1	Polinomial	5	0.75	-	-
20422	590	22	1,68%	0,001	Polinomial	4	0.25	-	-
2574	725	0	1,68%	10	Polinomial	6	0.25	-	-
8490	645	0	1,75%	0,01	Polinomial	5	-	-	-
7429	677	3	1,75%	1	Polinomial	5	-	-	-

Continua

Continuação

Nº Iterações	Nº VS	Nº VS *bound*	Erros %	C	Kernel	p	Constante k	Kapa κ	Sigma σ
2516	760	2	1,75%	1	Polinomial	6	0.75	-	-
7627	661	2	1,81%	0,0001	Polinomial	5	0.5	-	-
7454	675	1	1,81%	0,1	Polinomial	5	-	-	-
34032	571	5	1,81%	1	Polinomial	4	1	-	-
6966	658	3	1,81%	1	Polinomial	5	1	-	-
9013	667	4	1,81%	100	Polinomial	6	0.25	-	-
2555	755	2	1,87%	0,01	Polinomial	6	0.5	-	-
7300	660	4	1,87%	0,1	Polinomial	5	0.25	-	-
5903	669	0	1,93%	0,0001	Polinomial	5	1	-	-
2407	748	2	1,93%	0,001	Polinomial	6	0.25	-	-
1670	802	1	1,93%	0,01	Polinomial	6	0.75	-	-
1893	776	2	1,93%	0,1	Polinomial	6	0.75	-	-
1824	787	2	1,93%	0,1	Polinomial	6	1	-	-
20249	581	26	1,99%	0,001	Polinomial	4	-	-	-
21774	585	22	1,99%	0,001	Polinomial	4	1	-	-
2471	741	2	1,99%	0,01	Polinomial	6	0.25	-	-
1905	768	2	2,06%	0,001	Polinomial	6	1	-	-
6910	678	0	2,06%	0,1	Polinomial	5	0.75	-	-
42356	568	8	2,06%	10	Polinomial	4	1	-	-
1683	826	0	2,12%	0,0001	Polinomial	6	0.75	-	-
7164	674	3	2,12%	0,001	Polinomial	5	0.5	-	-
1846	806	1	2,12%	0,001	Polinomial	6	-	-	-
6060	666	2	2,12%	1	Polinomial	5	0.25	-	-
7595	652	0	2,12%	100	Polinomial	5	-	-	-
34026	572	3	2,18%	1	Polinomial	4	0.25	-	-
45016	563	1	2,37%	0,1	Polinomial	4	-	-	-
31526	569	6	2,37%	0,1	Polinomial	4	1	-	-
10464	631	5	2,37%	10	Polinomial	5	-	-	-
29878	592	2	2,37%	100	Polinomial	5	1	-	-
34950	529	59	2,49%	0,01	Polinomial	3	-	-	-
6804	673	3	2,49%	0,01	Polinomial	5	0.75	-	-
7880	659	7	2,56%	0,01	Polinomial	5	0.25	-	-
202139	501	20	2,56%	1	Polinomial	3	0.75	-	-
140243	531	17	2,56%	100	Polinomial	4	-	-	-
22148	582	21	2,62%	0,001	Polinomial	4	0.75	-	-
1983	773	0	2,62%	0,01	Polinomial	6	1	-	-
39158	557	5	2,62%	0,1	Polinomial	4	0.5	-	-
49130	557	12	2,62%	10	Polinomial	4	0.25	-	-
33584	582	3	2,68%	0,01	Polinomial	4	0.75	-	-
39124	574	8	2,68%	0,1	Polinomial	4	0.25	-	-
36253	568	7	2,68%	0,1	Polinomial	4	0.75	-	-
2550	750	0	2,68%	0,1	Polinomial	6	0.25	-	-
55403	575	9	2,68%	100	Polinomial	5	0.25	-	-
103410	539	12	2,74%	100	Polinomial	4	0.25	-	-
7470	652	0	2,80%	0,001	Polinomial	5	-	-	-
903	723	2	2,80%	1	Polinomial	7	-	-	-
7184	656	0	2,80%	10	Polinomial	5	0.75	-	-
321000	513	17	2,80%	1000	Polinomial	4	0.5	-	-
23004	580	21	2,87%	0,001	Polinomial	4	0.5	-	-

Continua

Continuação

Nº Iterações	Nº VS	Nº VS bound	Erros %	C	Kernel	p	Constante k	Kapa κ	Sigma σ
6924	651	4	2,87%	0,001	Polinomial	5	1	-	-
35290	566	6	2,87%	0,01	Polinomial	4	1	-	-
7295	654	0	2,87%	0,1	Polinomial	5	1	-	-
35236	535	54	2,93%	0,01	Polinomial	3	0.5	-	-
36416	522	53	2,93%	0,01	Polinomial	3	0.75	-	-
112590	538	9	2,93%	100	Polinomial	4	0.75	-	-
41502	560	2	2,99%	0,01	Polinomial	4	-	-	-
39081	567	5	2,99%	0,01	Polinomial	4	0.5	-	-
48645	561	8	2,99%	10	Polinomial	4	0.75	-	-
7840	473	168	3,12%	0,0001	Polinomial	4	1	-	-
40709	519	54	3,18%	0,01	Polinomial	3	1	-	-
321000	516	16	3,18%	1000	Polinomial	4	-	-	-
39946	580	1	3,24%	1	Polinomial	4	-	-	-
110128	534	11	3,24%	100	Polinomial	4	1	-	-
2478	765	0	3,24%	1000	Polinomial	6	0	-	-
35956	561	7	3,30%	0,01	Polinomial	4	0.25	-	-
6422	658	1	3,36%	0,001	Polinomial	5	0.75	-	-
321000	443	53	3,36%	1	Polinomial	2	1	-	-
43897	557	6	3,36%	10	Polinomial	4	0	-	-
7878	652	2	3,43%	0,0001	Polinomial	5	0	-	-
321000	498	20	3,49%	1000	Polinomial	3	0.5	-	-
321000	497	17	3,55%	100	Polinomial	3	0.5	-	-
6460	662	2	3,61%	0,01	Polinomial	5	1	-	-
234401	485	21	3,61%	10	Polinomial	3	0.75	-	-
321000	491	16	3,61%	100	Polinomial	3	1	-	-
7354	458	194	3,68%	0,0001	Polinomial	4	0.25	-	-
7581	444	196	3,68%	0,0001	Polinomial	4	0.5	-	-
321000	429	56	3,68%	1	Polinomial	2	0.5	-	-
86891	541	11	3,68%	100	Polinomial	4	0.5	-	-
321000	520	15	3,68%	1000	Polinomial	4	0.25	-	-
7231	442	200	3,74%	0,0001	Polinomial	4	0	-	-
321000	486	21	3,74%	1000	Polinomial	3	1	-	-
321000	494	15	3,74%	1000	Polinomial	4	0.75	-	-
247117	495	18	3,80%	1	Polinomial	3	0.25	-	-
6484	661	0	3,86%	0,0001	Polinomial	5	0.75	-	-
37753	520	54	3,86%	0,01	Polinomial	3	0.25	-	-
31486	574	9	3,86%	1	Polinomial	4	0.5	-	-
8805	659	1	3,86%	10	Polinomial	5	0.5	-	-
34017	395	18	3,86%	100	Polinomial	6	0.5	-	-
321000	531	17	3,93%	1000	Polinomial	4	1	-	-
137239	504	20	3,99%	0,1	Polinomial	3	-	-	-
182469	505	21	3,99%	10	Polinomial	3	0.5	-	-
298202	487	24	3,99%	100	Polinomial	3	0.25	-	-
7549	445	181	4,05%	0,0001	Polinomial	4	0.75	-	-
2185	754	0	4,05%	0,0001	Polinomial	6	0.25	-	-
158955	495	20	4,05%	0,1	Polinomial	3	0.75	-	-
303053	491	13	4,05%	1	Polinomial	3	0.5	-	-
321000	429	22	4,05%	100	Polinomial	2	0.5	-	-
321000	490	18	4,24%	100	Polinomial	3	-	-	-

Continua

Continuação

Nº Iterações	Nº VS	Nº VS *bound*	Erros %	C	*Kernel*	p	Constante k	Kapa κ	Sigma σ
317456	428	55	4,30%	1	Polinomial	2	-	-	-
309217	431	53	4,30%	1	Polinomial	2	0.25	-	-
321000	479	19	4,36%	1000	Polinomial	3	0.75	-	-
33446	561	9	4,42%	1	Polinomial	4	0.75	-	-
2163	747	0	4,42%	10	Polinomial	6	0.75	-	-
2359	752	0	4,49%	0,0001	Polinomial	6	0.5	-	-
226983	495	18	4,55%	10	Polinomial	3	0.25	-	-
2331	752	0	4,55%	10	Polinomial	6	0.5	-	-
140291	489	21	4,67%	0,1	Polinomial	3	0.5	-	-
235460	492	20	4,67%	1	Polinomial	3	-	-	-
321000	430	20	4,67%	100	Polinomial	2	1	-	-
70660	406	16	4,67%	1000	Polinomial	6	0.75	-	-
14675	608	9	4,74%	10	Polinomial	6	1	-	-
442	575	0	4,86%	1000	Polinomial	7	-	-	-
321000	495	18	4,92%	1000	Polinomial	3	0.25	-	-
192713	504	19	4,98%	10	Polinomial	3	-	-	-
180975	494	20	5,05%	10	Polinomial	3	1	-	-
321000	486	18	5,05%	1000	Polinomial	3	-	-	-
157365	498	22	5,11%	0,1	Polinomial	3	1	-	-
321000	494	20	5,11%	100	Polinomial	3	0.75	-	-
1943	795	1	5,17%	0,001	Polinomial	6	0.75	-	-
45837	563	5	5,17%	10	Polinomial	4	0.5	-	-
321000	418	21	5,17%	100	Polinomial	2	0.75	-	-
250459	481	17	5,30%	1	Polinomial	3	1	-	-
160398	495	20	5,36%	0,1	Polinomial	3	0.25	-	-
7329	662	0	5,55%	0,001	Polinomial	5	0.25	-	-
1254	852	0	5,67%	0,0001	Polinomial	6	1	-	-
321000	412	55	5,67%	1	Polinomial	2	0.75	-	-
321000	426	20	5,73%	100	Polinomial	2	-	-	-
206365	367	20	5,73%	100	Polinomial	5	0.5	-	-
363	508	0	6,11%	0,0001	Polinomial	7	0.25	-	-
321000	435	20	6,17%	100	Polinomial	2	0.25	-	-
44734	345	196	6,36%	0,1	Polinomial	2	-	-	-
321000	328	25	6,36%	1000	Polinomial	5	-	-	-
46179	338	206	6,48%	0,1	Polinomial	2	0.25	-	-
51254	342	206	6,48%	0,1	Polinomial	2	0.75	-	-
58520	352	18	6,48%	100	Polinomial	6	-	-	-
321000	415	21	6,48%	1000	Polinomial	2	0.5	-	-
50863	339	204	6,54%	0,1	Polinomial	2	1	-	-
321000	399	28	6,67%	1000	Polinomial	7	1	-	-
414	544	2	6,79%	0,1	Polinomial	7	0.25	-	-
46582	342	206	6,85%	0,1	Polinomial	2	0.5	-	-
321000	266	321	6,85%	1000	Sigmoidal	-	-0.75	0.01	-
321000	270	328	6,85%	1000	Sigmoidal	-	-1	0.01	-
6484	243	352	6,92%	0,001	Polinomial	3	-	-	-
7419	256	342	6,92%	0,001	Polinomial	3	1	-	-
321000	516	24	6,92%	1000	Polinomial	5	1	-	-
6919	245	338	6,98%	0,001	Polinomial	3	0.5	-	-
7203	254	336	6,98%	0,001	Polinomial	3	0.75	-	-

Continua

Conclusão

Nº Iterações	Nº VS	Nº VS bound	Erros %	C	Kernel	p	Constante k	Kapa κ	Sigma σ
6501	222	366	7,04%	0,001	Polinomial	3	0.25	-	-
7654	641	4	7,10%	100	Polinomial	6	0.75	-	-
421	558	0	7,35%	0,01	Polinomial	7	0.25	-	-
22098	608	4	7,41%	100	Polinomial	6	1	-	-
321000	419	18	7,41%	1000	Polinomial	2	0.75	-	-
321000	266	332	7,41%	1000	Sigmoidal	-	-0.5	0.01	-
321000	401	21	7,79%	10	Polinomial	2	0.5	-	-
321000	431	20	8,04%	1000	Polinomial	2	0.25	-	-
432	568	0	8,35%	0,1	Polinomial	7	-	-	-
321000	414	18	8,41%	10	Polinomial	2	-	-	-
321000	427	21	8,41%	1000	Polinomial	2	1	-	-
321000	415	20	8,54%	1000	Polinomial	2	-	-	-
321000	417	22	8,60%	10	Polinomial	2	0.25	-	-
435	580	0	8,72%	0,001	Polinomial	7	-	-	-
353	496	1	8,91%	0,001	Polinomial	7	0.25	-	-
321000	424	18	9,10%	10	Polinomial	2	1	-	-
379	533	0	9,16%	10	Polinomial	7	-	-	-
321000	427	19	9,47%	10	Polinomial	2	0.75	-	-
37792	415	28	9,53%	100	Polinomial	7	1	-	-
109896	540	17	9,72%	100	Polinomial	5	0.75	-	-

Fonte: A autora (2008)

APÊNDICE 5

DADOS *GERMAN*

Os dados *German* foram treinados com várias funções *kernel* e diferenciados parâmetros. Os parâmetros que apresentarem erros inferiores a 10% durante o treinamento encontram-se na tabela a seguir:

TABELA 21 – TREINAMENTO DOS DADOS *GERMAN*

Nº Iterações	Nº VS	Nº VS *bound*	Erros %	C	Kernel	p	Constante k	Kapa κ	Sigma σ
1459	65	590	0,00%	1	Gaussiano	-	-	-	0.001
1058	61	591	0,00%	1	Gaussiano	-	-	-	0.01
3044	71	585	0,00%	1	Gaussiano	-	-	-	0.1
3236	999	0	0,00%	10	Gaussiano	-	-	-	0.001
3005	999	0	0,00%	10	Gaussiano	-	-	-	0.01
2982	999	0	0,00%	10	Gaussiano	-	-	-	0.1
3236	999	0	0,00%	100	Gaussiano	-	-	-	0.001
3159	999	0	0,00%	100	Gaussiano	-	-	-	0.01
2503	999	0	0,00%	100	Gaussiano	-	-	-	0.1
25670	504	1	1,00%	0,001	Polinomial	4	-	-	-
23148	512	1	1,30%	0,001	Polinomial	4	0.5	-	-
28273	499	0	1,60%	0,01	Polinomial	4	1	-	-
27245	510	0	1,60%	0,1	Polinomial	4	0.75	-	-
26625	500	0	1,80%	10	Polinomial	4	0.5	-	-
54051	788	13	1,80%	10	Gaussiano	-	-	-	1
27836	511	0	1,90%	0,01	Polinomial	4	0.5	-	-
26043	515	0	1,90%	1	Polinomial	4	0.5	-	-
24953	524	0	1,90%	1	Polinomial	4	0.75	-	-
24270	525	0	1,90%	10	Polinomial	4	1	-	-
29074	503	0	1,90%	100	Polinomial	4	-	-	-
28929	501	0	2,00%	1	Polinomial	4	0.25	-	-
23499	505	0	2,00%	1	Polinomial	4	1	-	-
26018	518	0	2,00%	100	Polinomial	4	0.25	-	-
27086	515	0	2,10%	0,01	Polinomial	4	-	-	-
23409	510	0	2,10%	0,1	Polinomial	4	1	-	-
27933	509	0	2,10%	10	Polinomial	4	0.75	-	-
4347	610	0	2,10%	10	Polinomial	5	0.5	-	-
26535	509	1	2,20%	0,001	Polinomial	4	1	-	-
25248	507	0	2,20%	0,1	Polinomial	4	-	-	-
25984	510	0	2,20%	100	Polinomial	4	0.5	-	-
27611	501	0	2,30%	0,01	Polinomial	4	0.75	-	-
24755	502	0	2,30%	0,1	Polinomial	4	0.25	-	-
199800	445	0	2,30%	1	Polinomial	3	1	-	-
25186	517	0	2,40%	0,01	Polinomial	4	0.25	-	-
23313	516	0	2,40%	0,1	Polinomial	4	0.5	-	-
27892	502	0	2,40%	1	Polinomial	4	-	-	-

Continua

Continuação

Nº Iterações	Nº VS	Nº VS bound	Erros %	C	Kernel	p	Constante k	Kapa κ	Sigma σ
27322	508	1	2,50%	0,001	Polinomial	4	0.25	-	-
28668	493	0	2,50%	10	Polinomial	4	-	-	-
4312	599	0	2,60%	0,0001	Polinomial	5	0.75	-	-
28727	509	1	2,60%	0,001	Polinomial	4	0.75	-	-
5339	594	0	2,60%	0,01	Polinomial	5	-	-	-
4946	607	0	2,60%	0,1	Polinomial	5	0.5	-	-
4605	601	0	2,60%	100	Polinomial	5	0.25	-	-
26545	508	0	2,70%	100	Polinomial	4	0.75	-	-
5248	593	0	2,80%	10	Polinomial	5	-	-	-
4432	600	0	2,90%	0,0001	Polinomial	5	-	-	-
4515	252	498	2,90%	1	Gaussiano	-	-	-	1
4452	603	0	3,00%	0,01	Polinomial	5	0.75	-	-
4435	608	0	3,00%	1	Polinomial	5	0.75	-	-
199800	441	0	3,00%	10	Polinomial	3	0.75	-	-
25123	504	0	3,00%	100	Polinomial	4	1	-	-
4821	594	0	3,20%	0,001	Polinomial	5	0.25	-	-
199800	445	0	3,20%	100	Polinomial	3	1	-	-
4420	614	0	3,30%	0,01	Polinomial	5	1	-	-
199800	434	0	3,30%	1	Polinomial	3	0.25	-	-
4480	612	0	3,30%	1	Polinomial	5	-	-	-
4257	606	0	3,30%	100	Polinomial	5	1	-	-
4380	611	0	3,40%	0,0001	Polinomial	5	1	-	-
4590	608	0	3,40%	0,001	Polinomial	5	1	-	-
4886	598	0	3,50%	0,0001	Polinomial	5	0.5	-	-
4394	614	0	3,50%	0,1	Polinomial	5	0.75	-	-
199800	442	0	3,50%	10	Polinomial	3	0.25	-	-
199800	444	0	3,50%	10	Polinomial	3	0.5	-	-
4724	596	0	3,50%	10	Polinomial	5	0.75	-	-
60105	413	75	3,60%	0,01	Polinomial	3	0.5	-	-
199800	440	0	3,60%	1	Polinomial	3	0.75	-	-
170985	437	2	3,70%	0,1	Polinomial	3	0.25	-	-
199800	447	0	3,70%	1	Polinomial	3	0.5	-	-
191529	436	3	3,80%	0,1	Polinomial	3	-	-	-
4743	602	0	3,80%	0,1	Polinomial	5	-	-	-
4714	618	0	3,80%	1	Polinomial	5	0.25	-	-
199800	434	0	3,80%	100	Polinomial	3	0.5	-	-
4595	593	0	3,90%	0,001	Polinomial	5	0.5	-	-
60694	408	75	3,90%	0,01	Polinomial	3	0.75	-	-
66789	400	79	3,90%	0,01	Polinomial	3	1	-	-
199800	439	0	3,90%	10	Polinomial	3	1	-	-
5078	592	0	4,00%	0,0001	Polinomial	5	0.25	-	-
4299	619	0	4,00%	1	Polinomial	5	0.5	-	-
199800	439	0	4,10%	100	Polinomial	3	-	-	-
186948	440	3	4,20%	0,1	Polinomial	3	0.5	-	-
170346	445	1	4,20%	0,1	Polinomial	3	1	-	-
24538	502	0	4,20%	10	Polinomial	4	0.25	-	-
4639	615	0	4,30%	0,001	Polinomial	5	0.75	-	-
1150	681	0	4,30%	1	Polinomial	6	1	-	-
199800	444	0	4,30%	100	Polinomial	3	0.25	-	-

Continua

Conclusão

Nº Iterações	Nº VS	Nº VS bound	Erros %	C	Kernel	p	Constante k	Kapa κ	Sigma σ
58385	410	80	4,40%	0,01	Polinomial	3	-	-	-
4121	614	0	4,40%	0,1	Polinomial	5	0.25	-	-
60609	408	73	4,51%	0,01	Polinomial	3	0.25	-	-
196351	431	0	4,51%	10	Polinomial	3	-	-	-
4810	597	0	4,51%	100	Polinomial	5	0.5	-	-
199800	428	0	4,61%	100	Polinomial	3	0.75	-	-
5160	599	0	4,71%	100	Polinomial	5	-	-	-
180532	443	2	4,81%	0,1	Polinomial	3	0.75	-	-
1187	672	0	4,81%	10	Polinomial	6	-	-	-
4674	607	0	4,81%	100	Polinomial	5	0.75	-	-
10091	442	106	5,01%	0,0001	Polinomial	4	0.75	-	-
4761	597	0	5,01%	10	Polinomial	5	0.25	-	-
1006	698	0	5,01%	100	Polinomial	6	0.5	-	-
10012	414	127	5,11%	0,0001	Polinomial	4	-	-	-
9934	439	115	5,11%	0,0001	Polinomial	4	0.25	-	-
10297	424	115	5,21%	0,0001	Polinomial	4	0.5	-	-
199800	443	0	5,21%	1	Polinomial	3	-	-	-
4324	610	0	5,21%	10	Polinomial	5	1	-	-
11094	430	106	5,31%	0,0001	Polinomial	4	1	-	-
984	693	0	5,31%	0,01	Polinomial	6	1	-	-
199800	575	7	5,61%	100	Gaussiano	-	-	-	1
1242	662	0	5,81%	100	Polinomial	6	0.25	-	-
1090	681	0	5,91%	0,01	Polinomial	6	0.75	-	-
1027	691	0	6,21%	0,0001	Polinomial	6	0.75	-	-
4134	603	0	6,51%	0,1	Polinomial	5	1	-	-
1038	678	0	6,51%	100	Polinomial	6	1	-	-
1166	685	0	6,81%	0,0001	Polinomial	6	-	-	-
1166	695	0	6,81%	100	Polinomial	6	-	-	-
1264	672	0	6,91%	0,0001	Polinomial	6	0.25	-	-
1110	703	0	7,31%	1	Polinomial	6	0.5	-	-
4304	606	0	7,51%	0,01	Polinomial	5	0.5	-	-
1302	682	0	7,71%	1	Polinomial	6	-	-	-
4803	615	0	7,91%	0,01	Polinomial	5	0.25	-	-
1113	677	0	8,21%	0,1	Polinomial	6	0.5	-	-
4640	598	0	8,71%	1	Polinomial	5	1	-	-
1152	688	0	8,81%	0,1	Polinomial	6	-	-	-
199800	322	157	9,41%	1	Polinomial	2	0.75	-	-
1183	672	0	9,41%	100	Polinomial	6	0.75	-	-
1115	668	0	9,71%	10	Polinomial	6	0.25	-	-
1281	676	0	9,91%	0,001	Polinomial	6	0.25	-	-

Fonte: A autora (2008)

APÊNDICE 6

DADOS *AUSTRALIAN*

Os dados *Australian* foram treinados com várias funções *kernel* e parâmetros. Os parâmetros que obtiveram erros de treinamento inferiores a 10% encontram-se discriminados na tabela a seguir:

TABELA 22 – TREINAMENTO DOS DADOS *AUSTRALIAN*

Nº Iterações	Nº VS	Nº VS *bound*	Erros %	C	Kernel	p	Constante k	Kapa κ	Sigma σ
980	3	612	44,49%	0,0001	Polinomial	1	-	-	-
1124	43	604	0,00%	1	Gaussiano	-	-	-	0.001
1661	690	0	0,00%	10	Gaussiano	-	-	-	0.001
1664	690	0	0,00%	100	Gaussiano	-	-	-	0.001
15709	688	0	0,00%	100	Gaussiano	-	-	-	0.01
15342	685	2	0,15%	10	Gaussiano	-	-	-	0.01
6773	263	423	0,29%	1	Gaussiano	-	-	-	0.01
10335	98	449	3,19%	1	Gaussiano	-	-	-	0.1
35581	520	16	3,48%	10	Gaussiano	-	-	-	0.1
11775	209	16	5,65%	0,0001	Polinomial	6	0.75	-	-
24290	162	42	6,52%	0,001	Polinomial	5	1	-	-
22229	155	45	6,67%	0,001	Polinomial	5	0.5	-	-
35901	147	58	6,67%	0,01	Polinomial	4	0.75	-	-
5214	245	0	6,67%	0,1	Polinomial	7	-	-	-
30190	146	55	7,10%	0,01	Polinomial	4	0.25	-	-
74055	118	77	7,10%	0,1	Polinomial	3	1	-	-
11423	174	36	7,25%	0,0001	Polinomial	6	-	-	-
35218	148	56	7,25%	0,01	Polinomial	4	0.5	-	-
40106	154	56	7,25%	0,01	Polinomial	4	1	-	-
32184	149	58	7,39%	0,01	Polinomial	4	-	-	-
71739	120	76	7,39%	0,1	Polinomial	3	0.5	-	-
21041	201	0	7,54%	0,01	Polinomial	6	-	-	-
4487	121	97	7,68%	0,0001	Polinomial	5	0.25	-	-
75033	125	74	7,68%	0,1	Polinomial	3	0.75	-	-
138000	174	68	7,68%	10	Polinomial	2	0.75	-	-
138000	155	74	7,83%	10	Polinomial	2	-	-	-
5180	123	92	7,97%	0,0001	Polinomial	5	1	-	-
19966	165	38	7,97%	0,001	Polinomial	5	0.25	-	-
76194	121	76	7,97%	0,1	Polinomial	3	0.25	-	-
3217	234	0	8,12%	10	Polinomial	7	0.5	-	-
5042	122	92	8,26%	0,0001	Polinomial	5	0.75	-	-
23392	162	39	8,26%	0,001	Polinomial	5	0.75	-	-
138000	167	21	8,26%	0,1	Polinomial	4	-	-	-
4222	114	99	8,41%	0,0001	Polinomial	5	0.5	-	-
6491	99	110	8,41%	0,001	Polinomial	4	0.25	-	-

Continua

Conclusão

Nº Iterações	Nº VS	Nº VS bound	Erros %	C	Kernel	p	Constante k	Kapa κ	Sigma σ
138000	159	27	8,41%	1	Polinomial	3	0.75	-	-
3677	233	0	8,41%	100	Polinomial	7	0.5	-	-
11815	192	12	8,55%	0,0001	Polinomial	6	1	-	-
51197	116	77	8,55%	0,1	Polinomial	3	-	-	-
138000	166	24	8,55%	0,1	Polinomial	4	0.25	-	-
3285	247	0	8,55%	0,1	Polinomial	7	1	-	-
138000	165	31	8,55%	1	Polinomial	3	0.25	-	-
4421	122	94	8,70%	0,0001	Polinomial	5	-	-	-
7124	103	108	8,70%	0,001	Polinomial	4	1	-	-
3761	246	0	8,70%	0,001	Polinomial	7	0.5	-	-
138000	152	76	8,70%	10	Polinomial	2	0.25	-	-
138000	159	70	8,70%	10	Polinomial	2	0.5	-	-
138000	161	72	8,70%	10	Polinomial	2	1	-	-
18154	210	0	8,84%	100	Polinomial	6	0.25	-	-
67919	171	15	8,99%	0,01	Polinomial	5	-	-	-
17446	197	0	8,99%	0,1	Polinomial	6	0.75	-	-
6751	106	103	9,13%	0,001	Polinomial	4	0.5	-	-
67936	170	10	9,13%	0,01	Polinomial	5	0.5	-	-
138000	166	20	9,13%	0,1	Polinomial	4	0.75	-	-
138000	156	30	9,13%	1	Polinomial	3	0.5	-	-
4399	243	0	9,13%	10	Polinomial	7	0.25	-	-
6900	105	103	9,28%	0,001	Polinomial	4	0.75	-	-
66941	189	4	9,28%	0,01	Polinomial	5	1	-	-
138000	65	134	9,28%	1	Polinomial	2	1	-	-
9479	72	126	9,42%	0,01	Polinomial	3	0.25	-	-
17985	205	0	9,42%	0,01	Polinomial	6	0.25	-	-
85718	166	0	9,42%	0,1	Polinomial	5	0.75	-	-
138000	154	31	9,42%	1	Polinomial	3	1	-	-
17404	209	0	9,42%	1	Polinomial	6	0.75	-	-
95500	173	0	9,42%	100	Polinomial	5	0.5	-	-
11500	76	119	9,57%	0,01	Polinomial	3	1	-	-
19376	197	0	9,57%	0,1	Polinomial	6	0.5	-	-
138000	63	130	9,57%	1	Polinomial	2	0.75	-	-
6844	89	106	9,71%	0,001	Polinomial	4	-	-	-
8721	70	122	9,71%	0,01	Polinomial	3	0.5	-	-
138000	69	128	9,71%	1	Polinomial	2	0.25	-	-
17062	196	0	9,71%	10	Polinomial	6	0.5	-	-
16109	206	0	9,71%	100	Polinomial	6	0.75	-	-
14811	211	0	9,86%	0,001	Polinomial	6	1	-	-
8703	75	117	9,86%	0,01	Polinomial	3	-	-	-
67933	180	10	9,86%	0,01	Polinomial	5	0.75	-	-
138000	66	126	9,86%	1	Polinomial	2	0.5	-	-
22893	207	0	9,86%	1	Polinomial	6	-	-	-
3844	228	0	9,86%	1	Polinomial	7	0.5	-	-

Fonte: A autora (2008)